JCC&BD 2018
VI JORNADAS DE CLOUD COMPUTING & BIG DATA

Facultad de INFORMÁTICA
UNIVERSIDAD NACIONAL DE LA PLATA

UNIVERSIDAD NACIONAL DE LA PLATA

25 AL 29 DE JUNIO

Libro de Actas

JCC&BD 2018

VI Jornadas de Cloud Computing & Big Data

VI JORNADAS DE CLOUD COMPUTING & BIG DATA

VI Jornadas de Cloud Computing & Big Data

La Plata (Buenos Aires), 25 al 29 de junio de 2018

Organizador

Facultad de Informática - Universidad Nacional de La Plata

Libro de Actas: JCC&BD 2018 - VI Jornadas de Cloud Computing & Big Data; coordinación general de De Giusti Armando. - 1a ed. - La Plata: Universidad Nacional de La Plata. Facultad de Informática, 2018.

Libro digital, PDF

Archivo Digital: descarga y online
ISBN 978-950-34-1659-4

1. Actas de Congresos. 2. Computación. 3. Minería de Datos. I. Armando, De Giusti, coord. CDD 006.312

Comité Científico

Coordinador: Armando De Giusti (UNLP)

Aguilar, José (Universidad de Los Andes, Venezuela)
Abásolo, María José (UNLP)
Ardenghi, Jorge (UNS)
Balladini, Javier (UNCOMA)
Bria, Oscar (UNLP)
Castro, Silvia (UNS)
De Giusti, Laura (UNLP)
Denham, Mónica (UNRN)
Diaz, Javier (UNLP)
Doallo, Ramón (Universidad da Coruña, España)
Errecalde, Marcelo (UNSL)
Frati, Fernando Emmanuel (UNdeC)
Garcia Garino, Carlos (UNCuyo)
Gaudiani, Adriana Angélica (UNGS)
Gil, Costa (UNSL)
Guerrero, Roberto (UNLS)
Hasperué, Waldo (UNLP)
Igual Peña, Francisco Daniel (Universidad Complutense de Madrid, España)
Lanzarini, Laura (UNLP)
Leguizamón, Guillermo (UNSL)
Luque, Emilio (Universidad Autónoma de Barcelona, España)
Marín, Mauricio (Yahoo, Chile)
Marrone, Luis (UNLP)
Naiouf, Marcelo (UNLP)
Olcoz Herrero, Katzalin (Universidad Complutense de Madrid, España)
Olivas Varela, José Angel (Universidad de Castilla La Mancha, España)
Pardo, Xoan (Universidad da Coruña, España)
Piccoli, María Fabiana (UNSL)
Piñuel, Luis (Universidad Complutense de Madrid, España)
Pousa, Adrian (UNLP)
Printista, Marcela (UNSL)
Rexachs, Dolores (Universidad Autónoma de Barcelona, España)
Rodríguez, Nelson (UNSJ)
Saez, Juan Carlos (Universidad Complutense de Madrid, España)
Sanz, Victoria (UNLP)
Suppi, Remo (Universidad Autónoma de Barcelona, España)
Tirado, Francisco (Universidad Complutense de Madrid, España)
Touriño, Juan (Universidad da Coruña, España)
Zarza, Gonzalo (Globant)

Índice

Análisis funcional de la pila de software de E/S paralela utilizando *IaaS*

Diego Encinas[1], Sandra Mendez[2], Dolores Rexachs[3], Emilio Luque[3], Marcelo Naiouf[1], and Armando De Giusti[1]

[1]*Informatics Research Institute LIDI, CIC's Associated Research Center. National University of La Plata, 50 y 120, La Plata, 1900, Argentina.*
{dencinas, mnaiouf, degiusti}@lidi.info.unlp.edu.ar
[2]*High Performance Systems Division, Leibniz Supercomputing Centre (LRZ), Boltzmannstr. 1, Garching near Munich, 85748, Germany*
sandra.mendez@lrz.de
[3]*Computer Architecture and Operating Systems Department. Universitat Autònoma de Barcelona. Edifici Q Campus UAB. Bellaterra, 08193, Spain*
{dolores.rexachs, emilio.luque}@uab.es

Abstract

Las operaciones de Entrada/Salida en los centros de supercomputación siguen siendo un cuello de botella para el procesamiento en paralelo. Esto provoca que la capacidad de procesamiento de los distintos nodos que componen estos centros se vea limitada. Por este motivo, en el presente trabajo se propone estudiar y evaluar el rendimiento de las principales operaciones paralelas de E/S. Las ejecuciones se realizan utilizando dos tipos de clústeres (físico y virtual), empleando benchmarks específicos de E/S y sistemas de archivos. Además, se propone diseñar, desarrollar e incorporar pequeñas secciones de código a algunas de las funciones más importantes invocadas durante la ejecución de los benchmarks, de manera de obtener medidas de tiempo para cuantificar la ejecución en los distintos puntos del sistema de software de E/S.

Keywords: Cloud Computing, Pila de Software de E/S, PVFS2, Instrumentación de Código

1 Introducción

Los actuales sistemas de HPC proporcionan cada vez mayor capacidad de cómputo debido al rápido avance de las tecnologías de procesadores. Ese avance no es proporcional en el área de almacenamiento. Para reducir la diferencia entre el rendimiento computacional y el de almacenamiento, los actuales sistemas de E/S utilizan hardware de alto rendimiento por medio de redes de almacenamiento de alta velocidad y sistemas de discos en RAID para proporcionar E/S paralela. Para gestionar el hardware de E/S se usa la pila de software que está compuesta por capas que gestionan los diferentes módulos de la arquitectura.

La Figura 1 muestra dos visiones del sistema de E/S, la infraestructura hardware y de la pila de software, el rendimiento del sistema depende de la carga de trabajo (patrones de E/S de las aplicaciones) y de la configuración del sistema. Existe una gran variabilidad en el comportamiento global del sistema dependiendo de la configuración y de los patrones de acceso a la E/S generados por las aplicaciones, observar el impacto de la configuración del sistema requiere el uso de sistemas rápidamente reconfigurables [1].

Una operación de E/S debe atravesar la estructura de la pila de software para escribir/leer a/desde los dispositivos. El encargado de gestionar el acceso a los datos es el sistema de ficheros, en HPC para conseguir el paralelismo a nivel de E/S, se usan sistemas de ficheros paralelos. Cuando una aplicación se está ejecutando y realiza una operación de E/S, las librerías de E/S reciben la operación e interaccionan con el sistema de ficheros. Este tipo de sistemas multicapas trae una complejidad adicional que está relacionado con el análisis de rendimiento y de funcionalidad de toda la pila de software.

Una de las técnicas para evaluar el rendimiento del Sistema de E/S a nivel de libreria es el uso de *benchmarks* específicos como IOR, BT-IO, Madbench2, entre otros que generan diferentes patrones de acceso[2]. Para poder explicar el rendimiento observado se pueden utilizar herramientas de análisis de rendimiento para HPC, pero la mayoria de las herramientas que permiten analizar el rendimiento de la E/S lo hacen a nivel MPI-IO y de POSIX-IO. Por lo cual no es posible obtener información sobre cómo interactúan todas las capas de la pila de SW de E/S.

Entender la interaccion entre las capas de la pila de software es importante no solo para explicar el comportamiento observado, sino tambien para detectar cuellos de botellas y proponer tecnicas de optimizacion. Ademas, este tipo de informacion es fundamental para modelar y simular configuraciones del sistema de E/S para poder evaluar el impacto de las diferentes capas

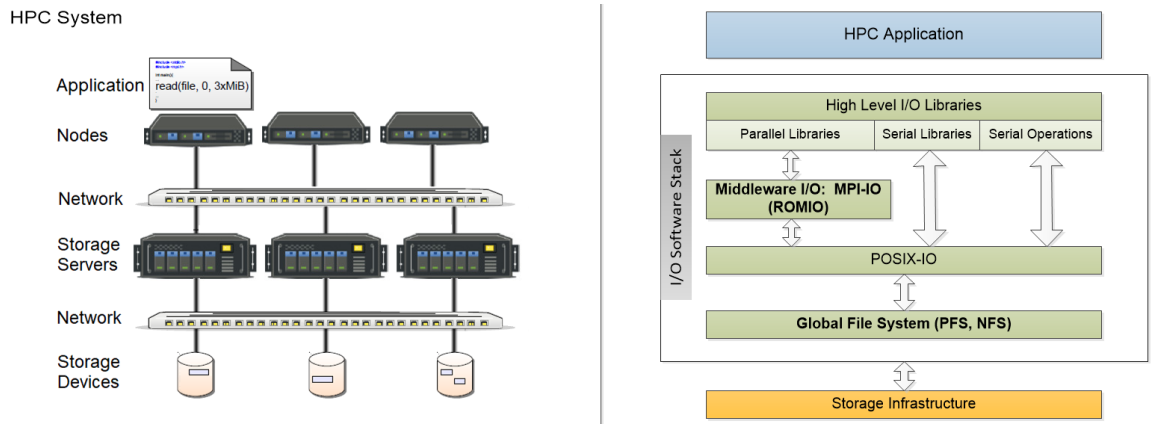


Figure 1: Sistema de HPC: infraestructura y pila de software de E/S

sin alterar sistemas en produccion.

En este artículo se presenta un análisis funcional de pila de software de E/S, por medio de la instrumentación de código en las capas de MPI-IO y del sistema de ficheros paralelo utilizando el servicio IaaS de Amazon. Para identificar las diferentes funciones que interactúan se ha optado por una técnica de instrumentación de código fuente en la capa MPI-IO y del sistema de fichero paralelo. Se ha seleccionado una plataforma Cloud, para desplegar un clúster de HPC y configurar la pila de software con todas las modificaciones necesarias. Se ha optado por una plataforma IaaS debido a que los sistemas de I/O en clusters fisicos requieren permisos de administrador y cualquier cambio afecta a todos los usuarios del cluster. Además, se pueden evaluar varias configuraciones de E/S de manera más rápida y practica sin afectar el normal funcionamiento de un sistema de HPC en producción. El trabajo está organizado de la siguiente manera, en la Sección 2 se presentan los conceptos relacionados al software utilizado. En la Sección 3 se describen las capas de la pila de software elegidas para explicar el trabajo. La Sección 4 muestra los experimentos realizados y los resultados encontrados. Finalmente, en la Sección 5 se presentan las conclusiones y trabajos futuros.

2 Conceptos relacionados

En esta sección se introducen algunos conceptos básicos relacionados a la pila de software de E/S.

2.1 Middleware

La Interfaz de Pasaje de Mensajes (MPI) es un sistema estandarizado y portable diseñado por un grupo de investigadores tanto del sector académico como del industrial para ser utilizado en una amplia variedad de arquitecturas de cómputo paralelo. Mientras que MPI-IO fue desarrollado en el Laboratorio Watson de IBM para proveer soporte de E/S paralela para MPI.

Los llamados de funciones de MPI-IO son muy similares a los de MPI, escribir archivos MPI es semejante a enviar mensajes MPI y leer archivos MPI es parecido a recibir mensajes MPI. En este trabajo se utilizó MPICH que es una implementación de alto rendimiento y ampliamente portátil del estándar MPI. MPICH [3] está siendo actualmente distribuido como software abierto, con una licencia libre y gratuita. Ha sido probado en múltiples plataformas, incluyendo Linux (en IA32 y x86-64), Mac OS/X (PowerPC e Intel), Solaris (32 y 64 bits) y Windows.

2.2 Benchmarks

Los benchmarks están diseñados para imitar un tipo particular de carga de trabajo en un componente o sistema. Los benchmarks sintéticos lo hacen creando programas especiales para imponer la carga de trabajo en el componente, mientras que los benchmarks de aplicación ejecutan programas del mundo real sobre el sistema. Los benchmarks de aplicación suelen ofrecer una mejor medición del rendimiento real de un sistema determinado, pero los sintéticos son útiles para probar componentes individuales, como un disco duro o un dispositivo de red. Idealmente, los benchmarks sólo deberían sustituir a las aplicaciones reales si la aplicación no está disponible o es demasiado difícil o costosa de ejecutar en un procesador o sistema determinado.

Se ha optado por dos de los benchmarks de E/S más populares para evaluar los rendimientos de las operaciones paralelas de E/S en las que se centrará este trabajo: uno sintético, conocido como IOR, y otro de aplicación, llamado BTIO.

2.3 Sistemas de Archivos Paralelos

Al constituir un clúster de cómputo de alto rendimiento, se tienen tres categorías principales de sistemas de archivos: NFS, los sistemas de red de área de almacenamiento (SAN) y los sistemas de archivos paralelos. El primero suele ser considerado fácil de

utilizar, pero el servidor NFS puede ser un lugar de fallo crítico y NFS no es correctamente escalable a través de grandes clústeres.

En cambio, entre las principales ventajas que puede proveer un sistema de archivos paralelo se puede mencionar un espacio de nombre global, escalabilidad y la capacidad de distribuir grandes archivos a través de múltiples nodos en un entorno de clúster, haciendo que un sistema de archivos de esta clase sea muy apropiado para subsistemas de E/S en HPC. Generalmente, un sistema de archivos paralelo implica un servidor de metadatos que contiene información sobre los datos en los nodos de E/S. Algunos sistemas utilizan un servidor particular para los metadatos, mientras otros distribuyen la funcionalidad del servidor de metadatos a través de los nodos de E/S. Algunos ejemplos de sistemas de archivos paralelos para clústeres de cómputo de alto rendimiento son PVFS y Lustre.

En este trabajo se ha optado por PVFS2 que es soportada por la Universidad Clemson y el Laboratorio Argonne. Aunque, puede mencionarse que Clemson desarrolló en 2008 una nueva rama llamada OrangeFS que incluye extensiones para soportar grandes directorios de pequeños archivos, mejoras de seguridad y capacidades de redundancia. Hacia el año 2011 OrangeFS se convirtió en la línea de desarrollo principal, aunque la mayoría de las actualizaciones son aplicables a ambas ramas [4].

3 Análisis funcional de la pila de software

Desde la petición por parte de la aplicación de usuario hasta que la misma llega a realizarse físicamente sobre el disco, es importante determinar los programas y módulos que utilizan cada una de las operaciones, esto permitirá decidir dónde incluir código adicional para determinar el tiempo de ejecución de las distintas funciones ante las diferentes operaciones de E/S. En la Figura 2 se muestra un esquema de la estructura analizada para realizar lo antes dicho.

En la Figura 2 se puede ver la estructura interna de MPICH con respecto a su implementación de MPI-IO y su vínculo con el sistema de archivos del cluster. ROMIO es una implementación portable y de alto rendimiento de MPI-IO, optimizada para patrones de acceso no contiguos, los cuales son comunes en las aplicaciones paralelas. Un componente clave de ROMIO, que permite una implementación tan portable de MPI-IO, es una capa interna de dispositivo abstracto para E/S paralela llamada ADIO (Abstract-Device Interface for I/O). Es el componente `ADIO` el que contiene los módulos que incluyen las funciones que envían las peticiones de E/S de las aplicaciones de usuario hacia el sistema de archivos paralelo. Es decir, la capa ADIO Common contienen las funciones comunes a todos los sistemas de archivos. Mientras que `AD_PVFS` contienen las funciones para implementar la interfaz MPI-IO en el sistema de archivos PVFS2.

3.1 Instrumentación

La instrumentación insertada en el código de los componentes de MPICH y del sistema de ficheros PVFS2 [5] ha sido utilizada para identificar las funciones invocadas durante la ejecución de una aplicación paralela. En particular las interacciones de alguna funciones que son ejecutadas durante el procesamiento de determinadas solicitudes como son las de apertura / cierre, escritura y lectura de archivos.

En la Figura 3 se muestran, por ejemplo, las funciones en la capa de *ROMIO* y de *System Interface* de PVFS2, detectadas durante la ejecucion de *IOR* y *BT-IO clase FULL.*

4 Evaluación Experimental

Plataformas como Amazon ofrecen distintos tipos de instancias de acuerdo al tipo de servicio que se adquiera. En este caso se desplegó un clúster de HPC haciendo uso del servicio gratuito de AWS.

4.1 Entorno Experimental

Existen varias herramientas para crear clústeres virtuales de manera automática en la plataforma EC2 de Amazon, algunas son StarCluster o CfnCluster. Pero en este caso era necesario desplegar un clúster de forma manual y con una pila de software instrumentada ya que el objetivo del trabajo es un análisis detallado de las diferentes capas que comprenden el sistema.

En la Figura 4 se pueden ver los entornos físicos y virtuales con las respectivas configuraciones.

Se creó un clúster con 5 nodos: un master y 4 nodos de cómputo. La incorporación de PVFS2 obligatoriamente debe ser llevada a cabo antes de instalar MPICH o los benchmarks para que estos programas puedan reconocerlo. Tanto en el clúster físico como en el virtual se utilizó SSH para ejecutar comandos e implementar las configuraciones. Se optó por asignar cuatro nodos de E/S o servidores de datos y un servidor de metadatos o storage. La descripción de las características de los clústeres se puede ver en la Tabla 1.

4.2 Aplicación

Los experimentos fueron diseñados para analizar el comportamiento del sistema frente a la escritura y lectura de tres tamaños de archivos: 1GB, 2GB y 4GB. Las razones de restringir a estos tamaños se deben al objetivo de no prolongar demasiado los tiempos de ejecución y al mismo tiempo no exceder el limitado espacio de almacenamiento ofrecido por el entorno del clúster virtual para almacenar datos. Para conseguirlo

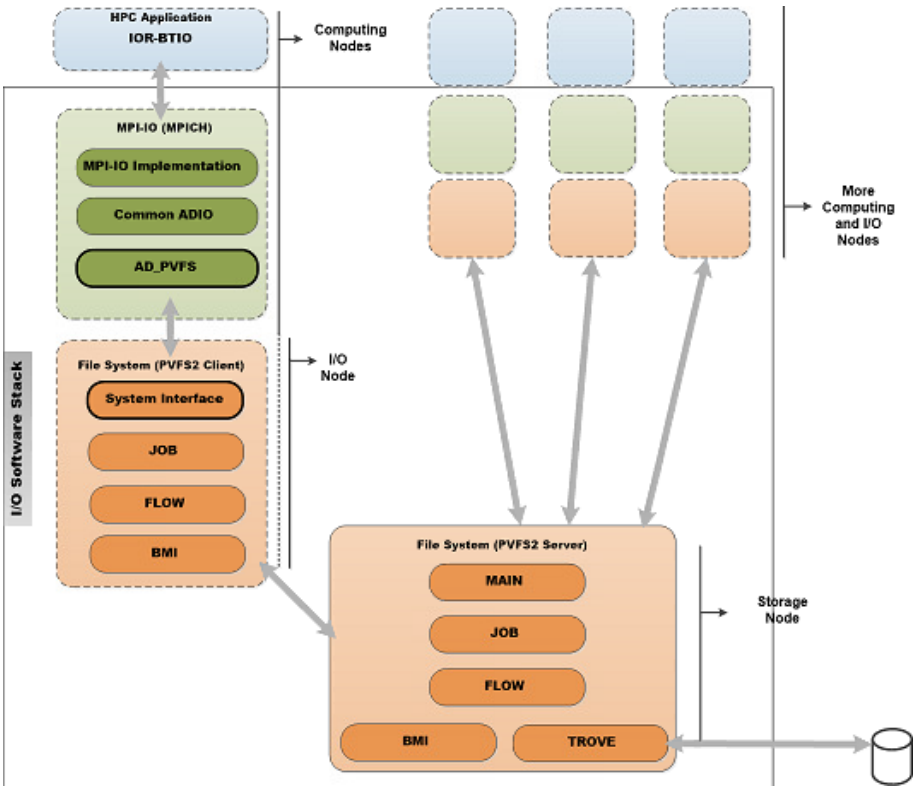


Figure 2: Capas analizadas de la pila de software de E/S

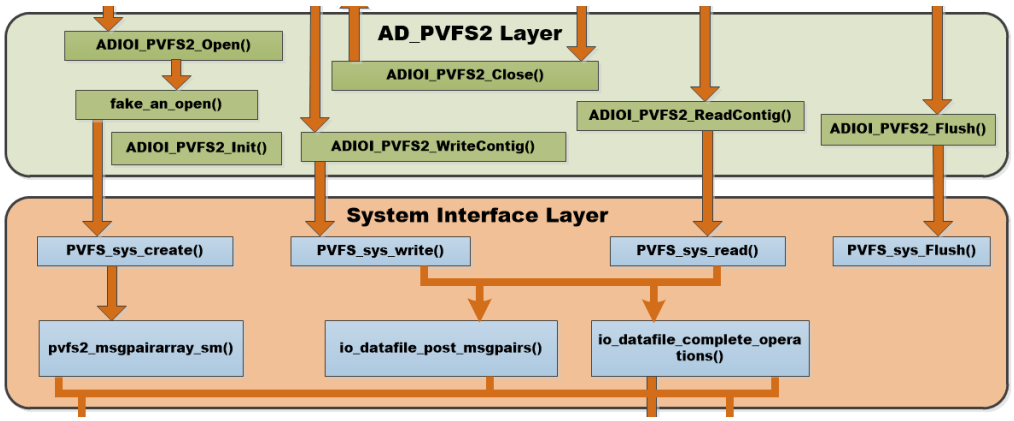


Figure 3: Funciones identificadas en la capa de MPI-IO y PVFS2 del lado del cliente

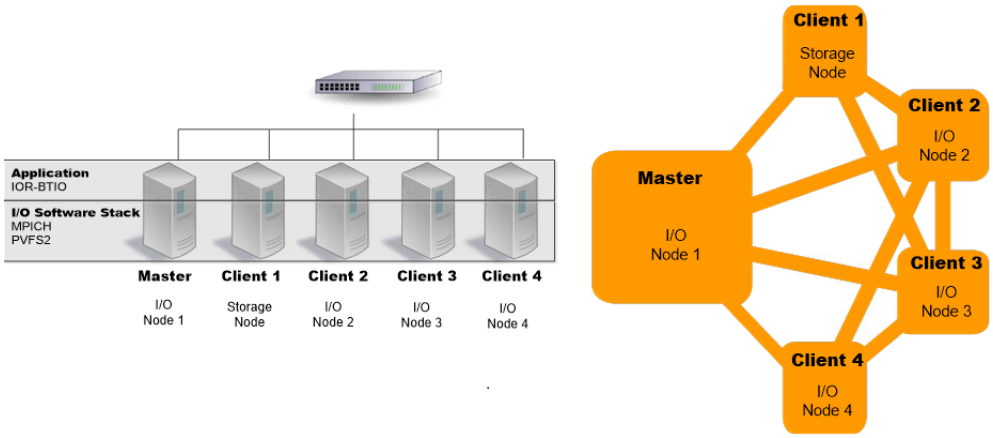


Figure 4: Clusters Físicos (figura en la izquierda) y Virtuales (figura en la derecha)

Table 1: Características de los clústeres utilizados

Componente	Clúster Físico	Clúster Virtual
Procesador	Xeon, i7, AMD	Xeon
RAM (por nodo)	8 GiB	1 GiB
Red de almacenamiento	1 Gbps Ethernet	Baja a Moderada
Número de Data Server	4	4
Número de MetaData Server	1	1
Librería MPI	MPICH versión 3.2.1	MPICH versión 3.2.1
Sistema de Archivos Global	PVFS 2 versión 2.8.2	PVFS 2 versión 2.8.2
Sistema Operativo	Red Hat Enterprise Linux	Red Hat Enterprise Linux

simplemente hubo que utilizar los parámetros adecuados en el caso de IOR. La línea de comandos utilizada fue:

```
mpirun -np NP ./IOR -a $API \
-t $TRANSFER_SIZE -b $BLOCK \
-F -o OUTPUT_FILE
```

En donde:

- `-np` representa el número de procesos MPI
- `-a` establece la interfaz o API de E/S
- `-t` define el tamaño en bytes del buffer de transferencia
- `-F` define que cada proceso trabaja sobre su propio archivo local
- `-b` define el número de bytes contiguos a escribir por proceso
- `-o` se establece el nombre completo del archivo de prueba

En el caso de BTIO se utilizó la clase W pero hubo que modificar el código fuente. Puntualmente se reemplazó la variable niter por el número de iteraciones necesarias para lograr el tamaño de archivo deseado. Ya que la clase seleccionada posee por defecto 200 iteraciones para escribir un archivo de 188 MB. A partir de esta relación se puede determinar fácilmente la cantidad de iteraciones requeridas para el tamaño de archivo buscado. En la Tabla 2 se pueden ver las configuraciones para la utilización de los dos benchmarks.

4.3 Resultados

Para mostrar los resultados se han tomado como ejemplo la capa `AD_PVFS2` que contiene las funciones para implementar la interfaz MPI-IO en PVFS2 y en este caso las funciones `ADIOI_PVFS2_WriteContig()` y `ADIOI_PVFS2_ReadContig()`. También la capa System Interface que es una interfaz del lado cliente que permite la manipulación de los objetos del sistema de archivos y en particular las funciones `PVFS_sys_write()` y `PVFS_sys_read()`. En la

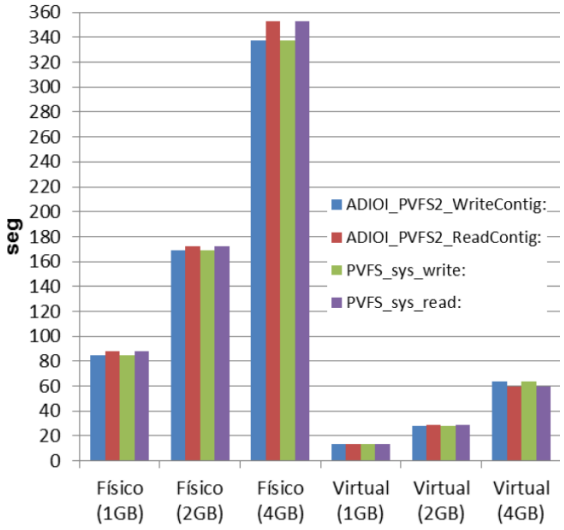


Figure 5: Métricas de IOR

Figura 5 se pueden ver los tiempos obtenidos con el benchmark IOR.

En la Figura 6 se pueden ver los tiempos obtenidos con el benchmark BTIO. Las diferencias con el clúster físico se deben principalmente a las distintas prestaciones de hardware conseguidas en los dos entornos de ejecución pero siguen la misma tendencia.

A pesar que los recursos hardware del clúster físico son ligeramente mejores a los del clúster virtual, en todos los casos el virtual presenta un rendimiento considerablemente superior. Se puede tomar esta situación como confirmación de que la velocidad de red es mucho más inferior en el clúster físico que en el virtual. Sumado al poco peso que poseen los accesos efectivos a disco sobre el tiempo de ejecución total del benchmark, provoca que el incremento en la complejidad de la red supera el beneficio que implica poder realizar múltiples lecturas y escrituras simultáneas en disco en lugar de hacerlas sólo de a una por vez.

Finalmente, se puede observar que con el clúster virtual se dispone de un sistema dedicado con el cual es posible realizar un análisis detallado de la pila de software de E/S. Esto permite definir la importancia temporal de los módulos o funciones de la pila relacionados al flujo de datos en los servidores de E/S

Table 2: Configuraciones seleccionadas para IOR y BTIO

Parámetro	**BTIO**	**IOR**
API para E/S (Interfaz)	MPI-IO	MPI-IO
Tamaño del buffer de transferencia	0,94 MB para 1GB 0,94 MB para 2GB 0,94 MB para 4GB	128 MiB para 1GiB 128 MiB para 2GiB 128 MiB para 4GiB
Tamaño del bloque de datos a escribir por proceso	256 MB para 1 GB 512 MB para 2GB 1 GB para 4GB	256 MiB para 1GiB 512 MiB para 2GiB 1 GiB para 4GiB

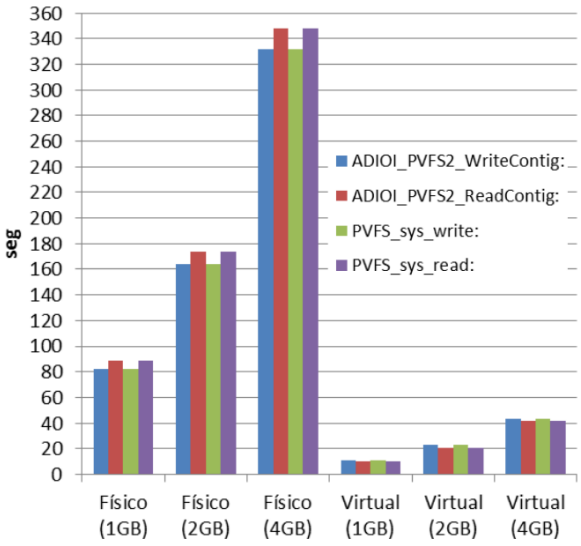


Figure 6: Métricas de BT-IO

como en los servidores de metadatos. La relevancia de conseguir medidas de tiempo en los componentes de la pila es fundamental ya que como se ha explicado anteriormente, distintos patrones de E/S o cargas de trabajo provocarán que el sistema se comporte de diferente manera.

5 Conclusión

El trabajo realizado en Cloud ha permitido identificar las funciones principales en la capa de MPI y del sistema de ficheros paralelo PVFS2 con mayor impacto en el comportamiento del sistema de E/S. Las métricas obtenidas en el clúster virtual no son similares a las encontradas en el clúster físico. Las diferencias se deben principalmente a las distintas prestaciones de hardware conseguidas en los dos entornos de ejecución. Sin embargo, los experimentos demuestran que se puede utilizar el entorno cloud para modificar y medir en el sistema ya que las medidas obtenidas representan el comportamiento de la pila.

Para poder usar esa información y esta técnica en sistemas en producción a nivel de usuario, se está trabajando en la implementación de un trazador (wrapper) para capturar las funciones de interés usando instrumentación dinámica, de manera que no sea necesario recompilar ninguna capa de la pila de software de E/S.

Agradecimientos

This research has been supported by the MICINN/MINECO Spain under contracts TIN2017-84875-P.

References

[1] P. Gomez-Sanchez, D. Encinas, J. Panadero, A. Bezerra, S. Mendez, M. Naiouf, A. D. Giusti, D. Rexachs del Rosario, and E. Luque, "Using AWS EC2 as Test-Bed infrastructure in the I/O system configuration for HPC applications," *Journal of Computer Science & Technology*, vol. Volumen 16, 11/2016 2016.

[2] R. Thakur, W. Gropp, and E. Lusk, "Data Sieving and Collective I/O in ROMIO," in *Proceedings of the The 7th Symposium on the Frontiers of Massively Parallel Computation*, FRONTIERS '99, (Washington, DC, USA), pp. 182–, IEEE Computer Society, 1999.

[3] MPICH, "Mpich overview," 2018.

[4] OrangeFS, "The orangefs project," 2018.

[5] T. PVFS2, "PVFS 2 File System Semantics Document," tech. rep., PVFS Development Team, 2015.

H-RADIC: The Fault Tolerance Framework for Virtual Clusters on Multi-Cloud Environments

Ambrosio Royo, Jorge Villamayor, Marcela Castro-León, Dolores Rexachs and Emilio Luque

CAOS – Computer Architecture and Operating Systems, Universitat Autònoma de Barcelona, Bellaterra (Cerdanyola del Vallès), Barcelona 08193, Spain
pabloambrosio.royo@e-campus.uab.cat,
{jorgeluis.villamayor,marcela.castro,dolores.rexachs,emilio.luque}@uab.cat

Abstract

Even though the cloud platform promises to be reliable, several availability incidents prove that they are not. How can we be sure that a parallel application finishes the execution even if a site is affected by a failure? This paper presents H-RADIC, an approach based on RADIC architecture, that executes a parallel application in at least 3 different virtual clusters or sites. The execution state of each site is saved periodically in another site and it is recovered in case of failure. The paper details the configuration of the architecture and the experiments results using 3 virtual clusters running NAS parallel applications protected with DMTCP, a very well-known distributed multi-threaded checkpoint tool. Our experiments show that the execution time was increased between a 5% to 36% without failures and 27% to 66% in case of failures.

Keywords: Cloud, Fault-Tolerance, High-Performance Computing, RADIC.

1. Introduction

We know that there aren't any computers small or big safe from failures, we have seen big cloud providers fail, Windows Azure had availability problems for the Olympic Games in 2012 [1], Amazon Web Services had been affected from the extreme weather [2] and by human errors [3], also Google Cloud Platform was attacked by a low-level software [4] or even Oracle Cloud's wide network error issue that was fixed by restarting the network [5].

Since communication with a cloud can be lost due to a wide range of possible errors, also causes loss of computational resources, power consumption and money; different authors have worked to prevent failures when there are parallel applications with message passing running on cloud environments; in the work of Villamayor et. al [6] where they propose Resilience as a Service (RaaS), a Fault Tolerant (FT) framework for High Performance Computing (HPC) applications running in a cloud, and Gomez et. al [7] propose a multi-cloud FT framework that was capable to continue working if any of the cloud sites fail and delivered the results on the due date. We have studied the previous work and took advantage of the elasticity (easy provisioning of "hardware") of cloud computing.

We propose to take the Redundant Array of Distributed Independent Controllers (RADIC) architecture and hierarchically scale it up to be applied to a fully automated, elastic FT framework for virtual clusters running on different cloud environments. Capable to protect applications running in private or public clouds from fatal fails like: loss of nodes during execution and loss of communication between clouds.

The next section presents some related state of the art work and the description of the RADIC architecture. In section 3, we will describe an overall detail of the H-RADIC architecture, fallowed up by section 4, a summary of the experiments done from the implementation of the H-RADIC architecture, after that section 5 with the conclusions, finishing with future work in section 6.

2. Background

In Japan, Bautista-Gomez et. al [8] proposed a low-overhead high-frequency multi-level checkpoint technique, that implement a three level checkpoint scheme that compensates for the overhead of the FT

by dedicating a thread of execution per node.

Another group of IEEE members [9] in the USA, set up an online two-level checkpoint model for HPC, one level deals with logical problems such as transient memory errors and the other one deals with hardware crashes like node fails, contributing with an online solution that determines the optimal checkpoint patterns and doesn't required the knowledge of the job length in advance.

Egwutuoaha et. al [10], from Australia, approached the problem by not relaying on spare node prior of a fail, aiming to reduce the time and cost of the execution on the cloud.

And of course, the work of Gomez et. al [7] and Villamayor et. al [6], that we mention in the introduction.

2.1. RADIC Architecture

The RADIC architecture is transparent and automatic, therefore the application doesn't have to be modified to apply it and there is no need of human intervention, also it is elastic since it has the ability to add new nodes whenever one crashes [11]. It consists on implementing FT for message passing applications, by intercepting and masking error which detect and manage errors instead of ending the application. Taking advantage of the hardware redundancy, RADIC needs at least three physical nodes to work so the application doesn't have to stop and start again. It works with two distributed software RADIC components [12]: Observers and Protectors; there is a protector running in each node and one observer for every process running in the application, as shown in Fig. 1. Additionally, a table named RadicTable is used to store the relation between nodes, observers and protectors which is updated by the RADIC Components.

Observer: When the application is launched, a software instance is created and attached to every processor used in the program, its job is to perform checkpoints and intercept the messages of its processor and send it to the Protector to store it.

Protector: It's in charge to request the observers to perform checkpoints and store the checkpoint files in its own Stable Storage (SS), and also to verify that the node that it's protecting is working and, in the case, that it fails launch the latest checkpoint.

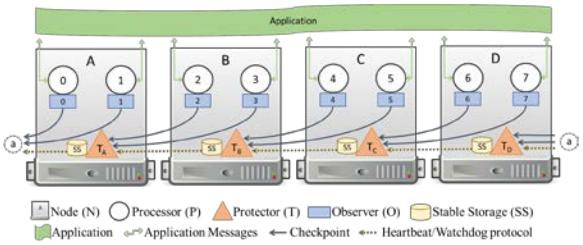


Fig. 1 - RADIC Architecture

RADIC defines four functions *Protection*, *Detection*, *Recovery* and *Error Masking*.

Protection: Observer O_3 is in charge of monitor Process P_3, then O_3 gets a Protector T_A located in a different node N_A. At some point T_A will ask O_3 to get the state of it process and send it to T_A to be stored in T_A´s Stable Storage (SS), this is called a checkpoint; checkpoints will be done periodically to keep on saving the state of the processes during a fault free execution. The moment that every T has its neighbor's checkpoint, the RADIC protection will be in place.

Detection: Node N_B faults when the process P_3 cannot communicate with another process P_2 in the same node, that's when O_3 sends the error message to T_A. Also, a node faults when there is no communication between nodes, that is, each protector (T_A) is in charge of detecting a possible failure in the neighbor Node (N_B). Each protector keeps a heartbeat/watchdog protocol with its neighbors Protectors, in this way a fault detection mechanism is implemented; in one hand, the Protector T_B is periodically sending heartbeats to T_C, and in the other hand, T_B is the watchdog of T_A, so if T_B loses communication with both neighbor protectors, T_B will destroy itself because it has been left alone, if T_A and T_C can't see T_B, then T_A will launch the recovery and error masking functions. Based on this function, RADIC needs at least 3 nodes to work properly.

Recovery: Protector T_A restarts/rolls-back the processes running in N_B, using the data saved in the SS from the last checkpoint, if the system has a spare node, the processes will be restarted in it, otherwise the processes are restarted in the node (N_A) that has the checkpoint.

Error Masking: After the recovery function, when the processes have been restarted, the Observers that got the error message hide the communication error caused from the node failure. The Protector T_A sends a message to the affected Observers ($O_{0\text{-}1}$ and $O_{4\text{-}5}$) with the new address where the processes have been recovered, also updates the RadicTable so communications are done as usual.

RADIC has been originally designed to protect a parallel application with message passing, recently it has been leveraged to work in cloud environments. In our work we scale up the architecture to be used for parallel application protection, running in several virtual clusters on multiple cloud environments.

3. H-RADIC Architecture

We propose a new protection level of RADIC, the Hierarchical RADIC (H-RADIC) architecture, an automated and elastic FT framework that protect a parallel application running in virtual clusters on

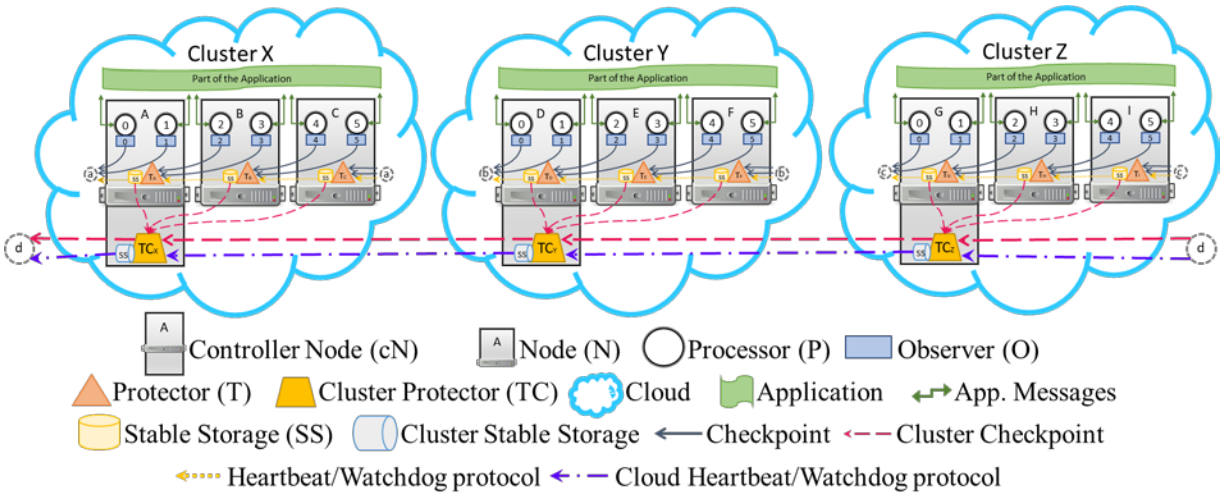


Fig. 2 - H-RADIC Architecture

cloud; requires at least three virtual clusters, each one of them protected with the RADIC architecture and located in different geographical sites so they don't share any of the physical resources, this will allow to identify nodes faults within and between the clusters.

The architecture works for parallel applications with a Message Passing Interface (MPI), that is, besides the regular RADIC architecture designed to protect from node fails, H-RADIC will protect applications from crashing in the event of multiple fails; when the virtual nodes fail, the physical nodes that the virtual are mounted on fail and/or whenever there is loss of communication. H-RADIC will perform a diagnostic and then management of the error.

Additionally to all the RADIC components, H-RADIC has an extra software component, the proTector of the Cluster (TC); on every cluster, a main node will be defined (controller Node - cN) to create the TC´s, which will be in charge of the communications between clusters, as it's shown Fig. 2.

3.1. H-RADIC functions

The H-RADIC architecture implements the RADIC functions in each cluster, that is, whenever there is a fault, the architecture tries to recover from it by applying the RADIC functions. H-RADIC functions will process the failures when the RADIC functions can't process them.

To guaranty the completion of the execution, H-RADIC transported the RADIC functions to be apply into programs running in multi-clusters environments and depict its functionality in the following functions:

Protection: Only while the execution is running fault free, checkpoints will be periodically done. Once that the Protectors (T) have the checkpoint of each of the process, the Cluster Protector (TC_Y) will be in charge of collecting the updates in the Stable Storage (SS) of every T in the Cluster (C_Y). Then, TC_Y must send it to the assigned Cluster Protector TC_X located in a different cloud to store the Cluster Checkpoint (multilevel checkpoint) at the Cluster SS. When every TC has a checkpoint of its neighbor C, the H-RADIC protection will be activated.

Detection: This function works the same way as the RADIC detection function; TC_X is in charge of identifying a possible failure in the neighbor Cluster C_Y, using a heartbeat/watchdog protocol. If C_Y does not answer, TC_X asks TC_Z to verify if there is an error with C_Y, if TC_Z confirms that there is no communication with C_Y, Recovery and Error Masking functions are triggered.

Recovery: Cloud Protector TC_K restarts/rolls-back the process running in C_L, using the data saved in the cluster SS from the last checkpoint, then TC_K checks if there is a spare cluster to lunch the checkpoint in it, otherwise creates new nodes in C_K to restart all the processes previously running in the failed C_L, then updates the RADIC Tables.

Error Masking: After the recovery function, when the processes have been restarted, the Cluster Protector hides the communication errors caused by the cloud failure, the Cluster Protector TC_K sends a message to the affected Protectors with the new address where the processes have been recovered, updating the H-RADIC Table and broadcasting this update to every TC in the architecture, then communications will be done as usual.

3.1.1. H-RADIC Recovery function' options

After a fault is detected, the recovery function has two options to restart a checkpoint; the first one checks if there are spare clusters available in another cloud, viewed in Fig. 3, the second one checks if there are spare nodes in the same cloud, as shown in Fig. 4, then the checkpoint files are sent to the nodes that are in the spare cluster.

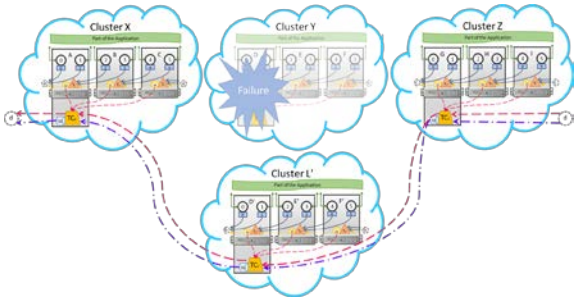


Fig. 3 - H-RADIC Recovery function - spare nodes/cluster in another cloud.

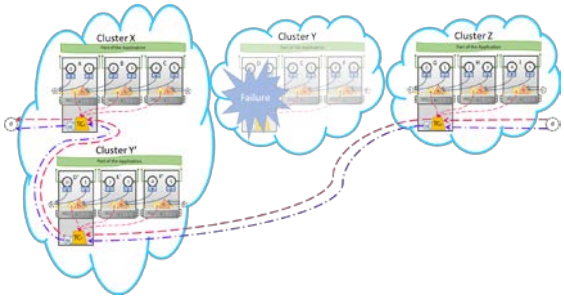


Fig. 4 - H-RADIC Recovery function - spare nodes/cluster in the same cloud.

The main difference between the two restart options is the way to store the checkpoints. In the case that there is a spare cluster available in other cloud (Fig. 3), H-RADIC will be working as usual, but if there are no spare nodes available in a third cloud, the checkpoint will be restarted in the spare nodes of the cloud that has the checkpoint. After the restart, the two clusters (X and Y' in Fig. 4) will send their checkpoints to Cluster Z and vice versa.

When the execution is working like the previous situation and if a new failure rises in one of the clouds, the three clusters will be together in one cloud, as shown in Fig. 5, and henceforth, all the checkpoints will be stored in one cloud storage.

Although this situation will leave the execution exposed to the same vulnerabilities, that lead us to develop this type architecture. Such vulnerabilities like failures in the cloud controller, the storage and the physical computer that the virtual nodes are mounted on. This vulnerability will now allow us to be able to guaranty availability.

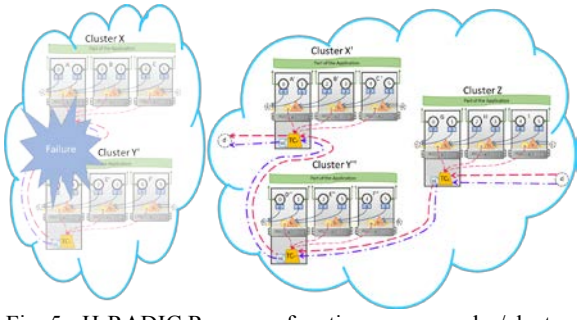


Fig. 5 - H-RADIC Recovery function - spare nodes/cluster in the same cloud and not more clouds left.

4. Experiments

To test the architecture, we mounted 3 clusters running on CentOS. Each one of them has a different NAS Parallel Benchmark [13] program compiled with an implementation of MPI named: MPICH [14]. H-RADIC will performing the checkpoints for this experiments in a coordinated approach, using the open source software package: Distributed Multi-Threaded CheckPointing (DMTCP) [15]. Each cluster has 3 nodes and every node have 8 processors, 24 processors per cluster.

The DMTCP allows to work with the checkpoints at an application layer of the OSI model, making it perfect for virtualized environments. Also, whenever a checkpoint is done, it automatically compresses it using gzip.

4.1. Pseudocode of H-RADIC

Before running the programs, they have to be compiled with the MPI implementation and also the inventory of nodes have to be depicted in the H-RADIC Table (Table 1); it's been established that each cluster will have a Cluster Protector, this will be the previous row in the table, and it´s expressed as (cluster-1), if the program is in cluster Y, the cluster protector will be cluster X. If the cluster it's in the first row (cluster X), then cluster protector will be the las row in the table (cluster Z), to complete a full loop.

Table 1 - H-RADIC TABLE

	Cluster	***Nodes***	***Controller Node***
1	clusterX	A, B, C	A
2	clusterY	D, E, F	E
3	clusterZ	G, H, I	G

All the logic described in the H-RADIC functions can be summarized in Algorithm 1.

Algorithm 1 - H-RADIC pseudocode

```
1:  // FUNCTIONS
2:  function protection (program, cluster, ckPTime){
3:    Send the program to the controllerNode in cluster
4:      launch dmtcpCoordinator with the program and ckPTime
5:      call detection(cluster, ckPTime)
6:      while
7:        perform checkPoint every ckPTime seconds
8:          send checkPoint to (cluster-1)
9:        if program ends
```

```
10:         broadcast succesfull end to all HRADIC Tables
11:         break while
12:   }
13:
14:   function detection (cluster, ckPTime){
15:     // Goes to the HRADIC Table and takes the next cluster in the table to protect
16:     establish watchdog protocol between cluster and (cluster+1)
17:     while
18:       get heartbeat from (cluster+1)
19:       send heartbeat to (cluster-1)
20:       if heartbeat/watchdogProtocol fails
21:         ask (cluster+2) to establish connection with (cluster+1)
22:         if the connection is established
23:           nothing happens
24:         else if (cluster+2) can't see (cluster+1)
25:           recovery(cluster+1, ckPTime)
26:   }
27:
28:   function recovery (cluster+1, ckPTime){
29:     search for spareCluster in other clouds or in the local cloud
30:     inizialize Nodes in spareCluster = Nodes in (cluster+1)
31:     send checkPoint files to nodes in spareCluster
32:     call errorMasking(spareCluster, cluster+1)
33:     // Call the protection fuction again, but this time run the checkpoint instead of the original program
34:     call protection(checkPoint,spareCluster,ckPTime)
35:   }
36:
37:   function errorMasking (spareCluster, cluster+1){
38:     remove (cluster+1) info. in the HRADIC Table
39:     add spareCluster info. in the HRADIC Table
40:     broadcast the update to all HRADIC Tables in all the clusters
41:   }
42:
43:   // MAIN
44:   // Launch all the protection functions in parallel
45:   execute protection(bt.C.16, clusterX, 60)
46:   execute protection(lu.C.16, clusterY, 80)
47:   execute protection(cg.C.16, clusterZ, 70)
```

4.2. Results

Calculating the optimal checkpoint interval is a controversial subject; for these experiments, all the programs were executed several times, this was done to identify the average time that each node needed to perform a checkpoint and move it to the different nodes and also to the cluster protector.

The experiment is measuring the time that it takes the application to finish its execution, in the following cases: 1) the application without applying H-RADIC but performing checkpoints, 2) the application with H-RADIC and without errors or failures and, 3) the application with H-RADIC and an induced error. Then by taking the increase percentage of time in 2) and 3), we got Fig. 6.

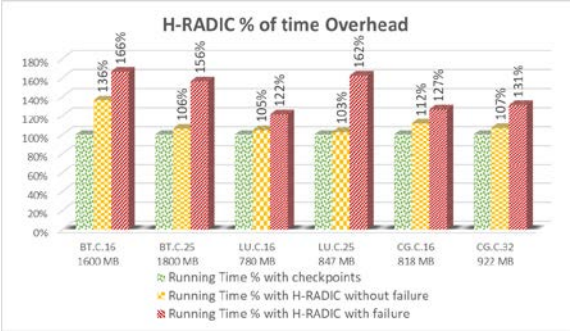


Fig. 6 - H-RADIC percentage of time overhead.

4.2.1. Overhead Breakdown

For the calculation of the overhead, we measured different time variables (as shown in Table 2): the first percentage shown in Fig. 6 is the application running with checkpoints as Eq (1), this time is considered the point of departure from where we are calculating the overhead.

$$T_{ckpt} = p + (ckpt \cdot n) \text{ (Eq. 1)}$$

In order to establish the RADIC and H-RADIC protection, all the checkpoints needed to be copied to the protector nodes and the cluster protector node respectively. This process is done at the background Eq (2) while the application keeps on executing. This process is represented in Fig. 2, by the checkpoint's and cluster checkpoint's lines. Since moving the files take some computational resources, it affects the program execution time, although its not directly in the critical path.

$$bkgrd = chkpSize \cdot protect \text{ (Eq. 2)}$$

The *Running Time percentage with H-RADIC without failure* in Fig. 6 is the time that the application took to execute considering the RADIC and H-RADIC protection in Eq (3).

$$T_{H-RADIC} = T_{ckpt} + bkgrd \text{ (Eq. 3)}$$

Finally, the third experiment, the *Running Time percentage with H-RADIC with failure* in Fig. 6 is the time that took the framework to restart the execution in a spare cluster Eq (4), considering the time to move

the checkpoint files and the time to restart the execution.

$$T_{H-RADIC\ \&\ Failure} = T_{H-RADIC} + move + restart$$
(Eq. 4)

Table 2 - Time's variables description.

Variable	Description
T_{ckpt}	Execution time with checkpoints
p	Program execution time
ckpt	Time to create a checkpoint
n	Number of checkpoints per execution
$T_{H\text{-}RADIC}$	Execution time with H-RADIC without failures
ckptSize	Checkpoint size in MB
protect	Time to move the ckpt to establish H-RADIC protection.
bkgrd	Unpredictable time impact on *p*
$T_{H\text{-}RADIC\ \&Failure}$	Execution time with H-RADIC with failure
move	Time to move the checkpoint files to the spare cluster
restart	Time to restart the application

In general, we can appreciate that the overhead caused by implementing H-RADIC and having 0 failures, does not really increase the program execution time, however if a failure rises, we can see the overhead percentage from as low as 22% to as high as 66% in time. This bottleneck is mainly due because of the time taken to move the checkpoint to the nodes in the spare cluster, since the time to restart the application once that the checkpoints where in the spare cluster is negligible.

5. Conclusion

The proposal in this paper is based on the RADIC architecture, that is capable of assure a successful execution of the application even when failures occur. We analyzed every component of the framework and identified the improvement areas.

By translating every RADIC concept to H-RADIC and taking them to a virtual multi-clusters level, we develop a FT framework capable of overcome fatal fails like the loss of a full site. The H-RADIC architecture fully implements RADIC and the new benefits from H-RADIC and it allows us to guarantee the completion of the execution in the event of errors like, loss of nodes, loss of communication between sites or general fails in several sites. It's a solution that can be implemented in virtual and non-virtual environments.

Considering that a traditional FT system, that consist on having one or two copies of a full site in another one, waiting for the main site to fail, with H-RADIC, by distributing the work load in different sites, the need of resources is way less that the ones needed for the traditional FT system.

Finally, the experiments show us that the overhead in most of the programs is reasonable and proves the theory behind the concept.

6. Future Work

Test the architecture implementing semi coordinated and no-coordinated checkpoints, to run a single program in several virtual clusters at the same time.

Develop the architecture in a way that if the performance is decreasing, it can request more resources live to the cloud provider, to assure competition before the due date.

Implement a mechanism that accelerates the transfer rate, by improving I/O, use incremental checkpoints and/or compress [16] the checkpoints even more.

Acknowledgements

The first author acknowledges the support from the National Science and Technology Council of Mexico (Consejo Nacional de Ciencia y Tecnología, CONACYT) that sponsored through a scholarship, and special thanks to all the team at CAOS and the UAB.

References

[1] B. Darrow, "Windows Azure outage hits Europe," 26-Jul-2012. [Online]. Available: https://gigaom.com/2012/07/26/windows-azure-outage-hits-europe/. [Accessed: 30-Mar-2018].

[2] O. Malik, "Severe storms cause Amazon Web Services outage," 29-Jun-2012. [Online]. Available: https://gigaom.com/2012/06/29/some-of-amazon-web-services-are-down-again/. [Accessed: 30-Mar-2018].

[3] "Summary of the Amazon S3 Service Disruption in the Northern Virginia (US-EAST-1) Region," *Amazon Web Services, Inc.* [Online]. Available: https://aws.amazon.com/message/41926/. [Accessed: 31-Mar-2018].

[4] "Google Cloud Status Dashboard." [Online]. Available: https://status.cloud.google.com/incident/storage/17002. [Accessed: 31-Mar-2018].

[5] J. Hult, "Oracle Cloud - unplanned outage - November 7, 2017," *JonathanHult.com*, 17-Nov-2017. .

[6] J. Villamayor, D. Rexachs, and E. Luque, "RaaS: Resiliance as a Service – Fault Tolerance for High Performance Computing in Clouds," p. Accepted, Mar. 2015.

[7] A. Gómez, L. M. Carril, R. Valin, J. C. Mouriño, and C. Cotelo, "Fault-tolerant virtual cluster experiments on federated sites using BonFIRE," *Future Gener. Comput. Syst.*, vol. 34, pp. 17–25, May 2014.

[8] L. Bautista-Gomez, S. Tsuboi, D. Komatitsch, F. Cappello, N. Maruyama, and S. Matsuoka, "FTI: High Performance Fault Tolerance Interface for Hybrid Systems," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, New York, NY, USA, 2011, p. 32:1–32:32.

[9] S. Di, Y. Robert, F. Vivien, and F. Cappello, "Toward an Optimal Online Checkpoint Solution under a Two-Level HPC Checkpoint Model," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 1, pp. 244–259, Jan. 2017.

[10] I. P. Egwutuoha, S. Chen, D. Levy, B. Selic, and R. Calvo, "Cost-oriented proactive fault tolerance approach to high performance computing (HPC) in the cloud," *Int. J. Parallel Emergent Distrib. Syst.*, vol. 29, no. 4, pp. 363–378, Jul. 2014.

[11] L. Fialho, G. Santos, A. Duarte, D. Rexachs, and E. Luque, "Challenges and Issues of the Integration of RADIC into Open MPI," in *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Springer, Berlin, Heidelberg, 2009, pp. 73–83.

[12] M. Castro-León, H. Meyer, D. Rexachs, and E. Luque, "Fault tolerance at system level based on RADIC architecture," *J. Parallel Distrib. Comput.*, vol. 86, pp. 98–111, Dec. 2015.

[13] "NAS Parallel Benchmarks," *NASA Advanced Supercomputing Division*. [Online]. Available: https://www.nas.nasa.gov/publications/npb.html. [Accessed: 23-May-2018].

[14] "MPICH | High-Performance Portable MPI," *MPICH*. [Online]. Available: https://www.mpich.org/. [Accessed: 02-Jun-2018].

[15] J. Ansel, K. Arya, and G. Cooperman, "DMTCP: Transparent checkpointing for cluster computations and the desktop," in *2009 IEEE International Symposium on Parallel Distributed Processing*, 2009, pp. 1–12.

[16] D. Tao, S. Di, X. Liang, Z. Chen, and F. Cappello, "Improving Performance of Iterative Methods by Lossy Checkponting," *ArXiv180411268 Cs*, Apr. 2018.

CLOUD ROBOTICs: Vehículo autónomo conectado a AWS

Manuel Costanzo[1], Marcos Boggia[1], Ismael Rodríguez[1] and Armando De Giusti[1,2]

[1] *Instituto de Investigación en Informática LIDI (III-LIDI), Facultad de Informática, Universidad Nacional de La Plata.*

[2] *Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina.*

{mcostanzo, mboggia, ismael, degiusti}@lidi.info.unlp.edu.ar

Resumen

El trabajo desarrollado presenta un avance de la publicación realizada en el XXIII Congreso Argentino de Ciencias de la Computación (CACIC 2017), con título "Cloud Robotics: Auto Rover 4WD y Cuadricóptero controlados remotamente desde AWS" [1], en el cual se incorporan algoritmos para procesamiento y reconocimiento de imágenes, algoritmos para simulación y determinación de ruta óptima al destino establecido, algoritmos para detección de obstáculos a evitar y espacios transitables, con bajo costo computacional.

Se presenta el despliegue de un sistema multi-robot, compuesto por un chasis de auto Rover 4WD con cámara de video y sensores integrados, como así también de una cámara aérea fija (simulando la captura de imágenes desde un cuadricóptero no tripulado – Drone), conectados al Cloud público de Amazon Web Services (AWS). Asimismo, se detallan los prototipos desarrollados, el protocolo de comunicación utilizado con el servicio de AWS IoT, y los algoritmos implementados para efectuar el control remoto del vehículo de cuatro ruedas no tripulado, con el fin de llegar a un destino de forma autónoma.

Palabras claves: Autonomous Cars, Amazon Web Services, Cloud Robotics, Internet of Things, Protocolo MQTT, Herramienta de programación visual Node-RED.

1. Introducción

Los avances en el paradigma de Cloud Computing han provocado un factor disruptivo de las TI en la industria tecnológica. Según el Instituto Nacional de Estándares y Tecnologías del Departamento de Comercio de los EEUU (NIST), como en varias publicaciones de diversos autores, se ha definido a Cloud Computing como: "un paradigma informático de cómputo distribuido, que proporciona grandes conjuntos de recursos virtuales (como ser hardware, plataformas de desarrollo, almacenamiento y/o aplicaciones), fácilmente accesibles y utilizables por medio de una interfaz de administración web. Estos recursos son proporcionados como servicios (del inglés, "as a service") y pueden ser dinámicamente reconfigurados para adaptarse a una carga de trabajo variable (escalabilidad), logrando una mejor utilización y evitando el sobre o sub dimensionamiento (elasticidad). El acceso a los recursos se realiza bajo la demanda de los usuarios, en base a un modelo de autoservicio" [2].

El modelo de Cloud Computing presenta las características y beneficios siguientes: recursos disponibles bajo demanda, escalabilidad y elasticidad, aprovisionamiento automático de recursos y autoservicio.

Dicho paradigma, brinda al menos tres modelos de despliegue: Cloud Público, Cloud Privado y Cloud Híbrido [3,4].

Por otro lado, considerando la gran influencia de la robótica en la sociedad y la diversidad de servicios ofrecidos por robots, nos encontramos que los mismos presentan grandes limitaciones en consumo de energía, poder de cómputo, capacidad de almacenamiento, toma de decisiones, tareas cognitivas, etc.

En el año 2010, comenzaron a surgir proyectos de investigación (ej: RoboEarth [5]), que integraban las tecnologías de Cloud con los sistemas de robots. Es así, que James Kuffner propone el concepto de Cloud Robotics, basado en combinar las tecnologías de robots con el paradigma de Cloud Computing [6].

La idea de Cloud Robotics, permite por medio de aplicaciones tratar los datos de los componentes de hardware del robot (sensores, actuadores/motores, cámaras, microcontroladores, memoria, etc.), sin importar las limitaciones de cómputo de las placas de desarrollo con microcontroladores y la capacidad de almacenamiento de las mismas [7]. En otras palabras, este concepto permite a los robots obtener resultados de tareas de cómputo intensivo, tales

como: procesamiento de imágenes, reconocimiento de voz, determinación de rutas, confección de mapas, acciones cognitivas, etc., sin tratamiento local, sino en el Cloud.
Este paradigma brinda la capacidad de establecer escenarios para sistemas de multi-robot, donde cada robot se integra de un hardware mínimo; una placa microcontroladora con conectividad WiFi permite al robot comunicarse con el Cloud y otros robots. Los datos de sensores y/o sistemas de adquisición de imágenes se procesan en el Cloud y los actuadores de cada robot llevarán a cabo las operaciones necesarias [8].
El propósito de investigación del presente trabajo es el despliegue de un sistema de multi-robot, inicialmente con dos robots conectados al Cloud Público de Amazon Web Services con el fin de simular el desplazamiento autónomo de un vehículo de 4 ruedas, como así también la gestión de una cámara aérea (simulando la captura de imágenes desde un cuadricóptero no tripulado - Drone), ambos conectados al Cloud público de AWS, donde corre una instancia de EC2 con algoritmos de procesamiento de imágenes que determinan la ruta que debe efectuar el vehículo autónomo para llegar a un destino establecido. Para alcanzar tales objetivos, se determinó el método de comunicación con el Cloud; se registraron los robots como dispositivos o "cosas" en el servicio de AWS IoT [9]; también, se confeccionaron los prototipos de cada robot y se incorporaron los algoritmos para procesamiento y reconocimiento de imágenes, algoritmos para simulación y determinación de ruta óptima al destino establecido, algoritmos para detección de obstáculos a evitar y espacios transitables, con bajo costo computacional.
El presente trabajo está estructurado de la manera siguiente: en la Sección 2, se introducen algunos conceptos elementales. A continuación, la Sección 3, presenta los prototipos confeccionados y sus componentes. En la Sección 4, se describe el trabajo experimental realizado. Por último, la Sección 5, expone los resultados obtenidos en relación a esta investigación.

2. Conceptos básicos

2.1. Amazon Web Services

Es una plataforma de Cloud Público, que provee "Infraestructura como servicio"; a través de la tecnología de virtualización, provee un gran conjunto de recursos de cómputo, como ser, almacenamiento y capacidad de procesamiento, que pueden ser solicitados a demanda, como así también, se adecuan dinámicamente en tamaño conforme la necesidad del consumidor. Los servicios más destacados de AWS son "Elastic Compute Cloud" (EC2), "Simple Storage Service" (S3) y en nuestro caso utilizaremos los servicios de "AWS Internet of Things" (AWS IoT) [10].

2.2. Internet of Things (IoT)

Es un nuevo paradigma cuya definición deriva de considerar "objetos" o "cosas" conectadas a Internet por medio de redes Wireless, sean estas de tecnología WiFi o 4G LTE, utilizando protocolos de comunicación estándar. Dichas "cosas" pueden considerarse etiquetas de identificación por radio frecuencia (RFID), sensores, actuadores, teléfonos móviles, placas de desarrollos, etc. [11].

2.3. Protocolo MQTT

Message Queuing Telemetry Transport [12] es un protocolo de comunicación ligero, especialmente diseñado para tolerar conexiones intermitentes y reducir los requisitos de ancho de banda de la red. Desde finales del año 2014, fue presentado como un estándar abierto OASIS [13] y se ha convertido en el protocolo por excelencia para IoT. Soporta comunicación segura con TLS. Maneja tres niveles de calidad de servicio: QoS 0: A lo sumo una vez la entrega del mensaje. QoS 1: Al menos una vez la entrega del mensaje. QoS 2: Exactamente una vez la entrega del mensaje.
Este protocolo, implementa la comunicación de mensajes por medio de la publicación/suscripción sobre un tópico específico (canal de comunicación), como se puede observar en la figura siguiente:

Fig. 1 MQTT: publicación/subscripción de mensajes.

2.4. AWS IoT

Es un servicio que proporciona AWS con el fin de conectar, administrar y operar grandes conjuntos de dispositivos o cosas. Ofrece mecanismos de conectividad y seguridad para la transmisión de datos. También, permite que los datos, una vez enviados a la plataforma, puedan ser procesados por las aplicaciones de análisis masivo de datos (Elastic MapReduce), análisis predictivo para aprendizaje automático (Amazon Machine Learning), almacenamiento en bases de datos, etc. [14].
AWS IoT permite conectar fácilmente dispositivos al Cloud y a otros dispositivos. El servicio es compatible con los protocolos: HTTP, WebSockets y MQTT. Amazon ha optado por este último, por ende AWS IoT utiliza la especificación del protocolo MQTT v.3.1.1 con QoS 0 y 1.

Cada dispositivo debe ser registrado como una "cosa" en AWS IoT, para lo cual se emitirá un certificado y un par de llaves privada-pública para el mismo. El certificado y la llave privada, junto con el certificado de la Entidad Certificantes (CA) de Amazon, deberán ser instalados en el dispositivo con el fin que este pueda conectarse al servicio de AWS IoT previa autenticación necesaria.

2.5. Node-RED

Es una herramienta de programación visual, que permite programar algoritmos basados en flujos para IoT, sin la necesidad de escribir código. Provee de un editor de flujo basado en navegador WEB, que brinda una amplia paleta de nodos con diversa funcionalidad. El flujo se confecciona conectando los nodos entre sí. La funcionalidad de cada nodo está desarrollada sobre Node.js. Los flujos creados son almacenados utilizando JSON, lo cual facilita la importación o exportación para compartir los mismos [15].

3. Prototipos y componentes

Para el presente trabajo se ha ensamblado un chasis de robot Rover de cuatro ruedas, que simula un vehículo tradicional de propulsión trasera, es decir, con un motor que ejerce la transmisión del movimiento sobre el eje trasero; también, cuenta con la capacidad de orientar las ruedas delanteras para la dirección del mismo. Igualmente, se incorpora una cámara de video integrada para la detección de obstáculos de forma local al vehículo; y se utiliza una placa micro-controladora de bajo costo y un módulo de expansión compatible, para controlar el motor de la tracción trasera y un sensor de ultrasonido, como se detallan a continuación:

3.1. Chasis Rover 4WD con servo S3003

Es un chasis compuesto de dos capas de acrílico, con un motor de 3 a 24v DC y velocidades con carga de 258 a 347 rpm/min, cuatro ruedas y un servo S3003 que permite direccionar el chasis con giros de hasta 45 grados. Sus dimensiones son: 248mm (L) x 146mm (W) x 70mm (H).

Fig. 2 Chasis Rover 4WD.

3.2. Microcontroladora Raspberry Pi 3

Placa de bajo costo de desarrollo que brinda conexión WiFi, almacenamiento en memoria MicroSD de 32Gb y soporta alimentación de energía de 5v DC [16].

Fig. 3 Raspberry Pi 3.

3.3. RaspiRobot Board V3

Es un módulo de expansión [17] compatible con Raspberry Pi 3, que brinda la interfaz de conexión entre la placa de desarrollo y los motores; incluye una fuente de alimentación conmutada para el suministro de energía, tanto de la placa Raspberry Pi como de los actuadores.

Fig. 4 Raspirobot Board V3.

3.4. Sensor Ultrasónico HC-SR04

El HC-SR04 es un sensor de distancias por ultrasonidos capaz de detectar objetos y calcular la distancia a la que se encuentra en un rango de 2 a 450 cm; cuantifica el delay de la señal, es decir, el tiempo que demora la señal en regresar el eco.

Fig. 5 Sensor ultrasónico HC-SR04.

3.5. Raspberry Pi Camera Board

Pi Camera V2.1, es un módulo de video alta calidad de 8MP con sensor de imagen Sony IMX219 y un lente de foco fijo. Es capaz de tomar imágenes

estáticas de 3280 x 2464 pixeles como así también, capturar vídeo de 1080p30, 720p60 y 640x480p90.

Fig. 6 Raspberry Pi Camera Board V2.1.

3.6. Fuente conmutada Step-Down

Fuente de alimentacióncon regulador Step-Down DC-DC XL4015, que brinda una salida regulada de 1.25 a 36V.

Fig. 7 Fuente DC Step-Down.

3.7. Acelerómetro y Giroscopio de 3 ejes

EL módulo Acelerómetro MPU tiene un giroscopio de tres ejes con el que podemos medir velocidad angular y un acelerómetro también de 3 ejes con el que medimos los componentes X, Y y Z de la aceleración. Utilizamos el sensor de giroscopio para que el robot pueda posicionarse en la dirección correcta.

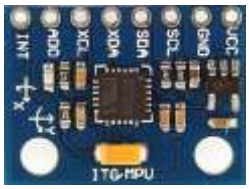


Fig. 8 Módulo MPU-6050.

4. Trabajo experimental

El trabajo de investigación consistió en desarrollar una plataforma web que permita controlar remotamente, vía internet, el sistema de multi-robot que emula un vehículo autónomo de 4 ruedas y una cámara de video de un Drone.
El escenario está compuesto por el Robot Rover 4WD, que posee una identificación por medio de la figura geométrica de un "cuadrado blanco con bordes negros". Así también, se dispone de un punto de destino identificado por la figura geométrica "circulo blanco con bordes negros" para la detección aérea y un "cuadrado blanco con bordes negros" para la detección desde el Robot Rover 4WD. Por otro lado, los objetos obstáculos están identificados por las figuras geométricas de "círculos rojos". Quien se encarga de identificar cada componente es el Cloud, por medio de las imágenes provenientes de la cámara aérea que emula a un Drone. Una vez identificados el Robot Rover 4WD y el punto destino sobre el escenario, se calcula el camino óptimo para que el vehículo se desplace hacia el punto desinto.
A continuación se describen los procedimientos para el procesamiento de imágenes y la determinación del camino óptimo.

4.1. Procesamiento de imágenes

El procesamiento de imágenes se realiza en una instancia de EC2 [18] en el Cloud, utilizando la librería OpenCV [19] y lenguaje de programación Python [20]. Se implementó un algoritmo que permite detectar e identificar sobre una imagen de captura aérea, las coordenadas de ubicación del Robot Rover 4WD con respecto al punto de destino; tomando como base dichas coordenadas se realiza el cálculo de la distancia entre el vehículo y el destino, como así también, los grados de diferencia entre los mismos.

4.1.1. Cámara aérea

La cámara aérea emula la captura de video que puede realizar un Drone. Dentro del escenario experimental se ubicó a una altura de tres metros sobre el nivel de la superficie del Robot Rover 4WD, enfocando el escenario completo. Dicha cámara está integrada a una placa de desarrollo Raspberry Pi 3.
Cuando el Robot aéreo recibe una señal vía el protocolo MQTT, desde el Cloud, solicitando una captura de imágen, el dispositivo realiza la captura de video en una resolución de 640x480 pixels, luego, disminuye la calidad en un 80% y realiza una compresión de la misma en base64, para por último enviarla vía MQTT a la instancia EC2, con el fin que el Cloud pueda realizar el reconocimiento y ubicación del Robot Rover 4WD y el punto de destino.
Todo el procesamiento mencionado se implementó con el lenguaje de programación visual Node-RED.

4.1.2. Cámara local

El Robot Rover 4WD, posee de forma integrada una cámara de video que permite realizar capturas de forma local al vehículo.
Cuando llega una notificación del Cloud, se efectua una captura de video y se realiza el procesamiento de compresión detallado anteriormente en la cámara aérea y se envía la imagen a la instancia EC2 en el Cloud.
A diferencia de la imágen capturada por la cámara aérea, en este caso, el punto destino es identificado por el reconocimiento de la figura geométrica "cuadrado blanco con bordes negros".

4.1.3. Instancia EC2

Cuando la instancia EC2, recibe una imagen, vía MQTT, se procede a identificar las figuras geométricas en la imágen. En el caso de la imagen capturada por la cámara aérea, como ya se ha mencionado, se tiene que detectar un "cuadrado blanco con bordes negros" para identificar al Robot Rover 4WD; y la figura geométrica de un "círculo blanco con bordes negros" para identificar al punto de destino. Por otro lado, en el caso de la imagen capturada por la cámara local del Robot Rover 4WD, se debe identificar la figura geométrica de un "cuadrado blanco con bordes negros" para identificar al punto de destino.
Una vez detectadas las figuras, se prosigue a calcular la distancia entre el vehículo y el destino. Previamente a este cálculo, es necesario haber calibrado las cámaras. Debido a que la visión de las cámaras es distinta, ya que la cámara aérea puede observar el escenario completo, en cambio, la cámara local sólo el punto de destino, la calibración de las mismas también difiere.
La figura 9 presenta el flujo de procesamiento de una imagen.

Fig. 9 Flujo de procesamiento de imágenes.

4.1.4. Calibración cámara aérea

El proceso de calibración se basa en realizar cálculo de la medida del ancho (en centímetros) de la figura geométrica del "cuadrado blanco con bordes negros" del Robot Rover 4WD. El objetivo de la calibración es determinar cuántos "centímetros" hay en un pixel. Con esta referencia, se podrá conocer la distancia en centímetros entre cualquier par de pixels.
Para lograr tal calibración, se calcula la distancia Euclidiana [21] entre dos extremos de la figura geométrica del cuadrado, y se divide por el ancho de referencia, como se puede observar en el algoritmo 1.

Algoritmo 1 Distancia Euclidiana

```
1: # Librería para calcular la distancia
   euclidiana en PyTHON
2: from scipy.spatial import distance as dist
3: def get_distance(p1, p2, reference):
4:   return dist.euclidean(p1, p2) / reference
```

Ejecutado el algoritmo 1, se logra obtener el conocimiento de cuántos pixels forman un "centímetro".
Dicha información es útil para la determinación de la distancia en "centímetros" entre el Robot Rover 4WD y el punto de destino; ahora alcanza con realizar el cálculo las coordenadas (x,y) del centro de las figuras geométricas, e invocar a la función del algoritmo 1, reemplazando los parámetros p1, p2 y reference por las coordenadas del Robot Rover 4WD, las del punto destino y el resultado de la calibración respectivamente.

4.1.5. Calibración cámara local

Este proceso de calibración [22] difiere del anteriormente mencionado, ya que no calcula la distancia entre objetos sobre una imágen, por el contrario, requiere conocer la distancia entre la cámara local y la figura geométrica "cuadrado blanco con bordes negros" del punto destino. Para esta calibración es necesario contar con dos datos: el ancho de la figura geométrica y la distancia en que se encuentra el Robot Rover 4WD con respecto de la figura al momento de la calibración; con dichos datos, se procede a realizar el cálculo de la longitud focal de la cámara local por medio de la ecuación 1:

$$\text{Longitud\_Focal} = \frac{WFigPX * Distancia}{WenCM} \quad \text{(ecuación 1)}$$

Donde:

WFigPX: es el ancho de figura en Pixels
WenCM: Ancho en centímetros.

El cálculo de la distancia focal se obtiene tras la ejecución del algoritmo 2.

Algoritmo 2 Longitud Focal

```
1: # Librería de Opencv
2: import cv2
3: def focal_length(contour, width, distance):
4:   marker = cv2.minAreaRect(contour)
5:   return (marker[1][0] * distance) / width
6: #marker[1][0] retorna el ancho en pixels
   de la figura geométrica.
```

Una vez obtenida la distancia focal, se procede a calcular la distancia entre la cámara y cualquier

objeto en la imagen local, por medio de la ecuación 2:

$$\text{Dist\_Objeto} = \frac{WenCM * Longitud\_Focal}{WFigPX} \text{ (ecuación 2)}$$

Donde:

WenCM: Ancho en centímetros.
WFigPX: es el ancho de figura en Pixels

La distancia al objeto se obtiene ejecutando el algoritmo 3.

Algoritmo 3 Distancia Objeto

```
1:  # Librería de Opencv
2:  import cv2
3:  Def distance_from_object_to_camera(contour, width, focal_length):
4:    marker = cv2.minAreaRect(contour)
5:    return (width * focal_length) / marker[1][0]
```

4.2. Planificación de camino óptimo

Una vez finalizada el procesamiento de las imágenes y realizado el cálculo de distancia entre el Robot Rover 4WD y el punto de destino, se procede a determinar el camino óptimo.

Un camino está compuesto por un conjunto de movimientos que guiarán al vehículo autónomo desde la posición donde se encuentra, hacia el punto de destino, evitando que colisione con los obstáculos que se presenten de por medio. Es necesario aclarar que si bien el algoritmo de planificación, que se detalla a continuación, contempla la identificación y el rodeo de los obstáculos, este trabajo se limita a que el Robot Rover 4WD alcance el punto de destino sin obstáculos de por medio.

El Robot Rover 4WD puede realizar movimientos de giro en grados a derecha e izquierda (de 0 a 180 grados y de 0 a -180, respectivamente), como así también, movimientos para avanzar una determinada distancia en centímetros.

La técnica de planificación elegida es la denominada RRT (Rapidly Exploring Random Trees) [23]. Este algoritmo se basa en la construcción de un árbol de configuraciones que crece buscando a partir de un punto origen en forma aleatoria, como se puede observar en el algoritmo 4.

Previamente, se definen algunas variables:

- self.node_list: es el árbol generado
- self.start, self.end: son las coordenadas de inicio y de fin respectivamente
- self.rand_y, self.rand_x: es el espacio de configuraciones
- self.expand_dis: es la distancia en el que los nodos se expanden. En nuestro caso utilizamos la distancia euclidiana calculada en la calibración de la cámara aérea, por lo que los nodos están separados a un centímetro de distancia.

Algoritmo 4 Algoritmo RRT

```
1:  def planning(self):
2:      # Flag que indica si encontró el destino.
3:      goal_flag = False
4:      # Lista de nodos del path.
5:      self.node_list = [self.start]
6:    for i in range(MAX_ITER):
7:        # Búsqueda aleatoria
8:        if random.randint(0, 100) > self.goal_sample_rate:
9:            rnd = [random.uniform(self.rand_x[0], self.rand_x[1]), random.uniform(
10:               self.rand_y[0], self.rand_y[1])]
11:       else:
12:           rnd = [self.end.x, self.end.y]
13:       # Busca el nodo más cercano.
14:       nind = self.get_nearest_list_index(self.node_list, rnd)
15:       nearest_node = self.node_list[nind]
16:       theta = math.atan2(rnd[1] - nearest_node.y, rnd[0] - nearest_node.x)
17:       new_node = copy.deepcopy(nearest_node)
18:       new_node.x += self.expand_dis * math.cos(theta)
19:       new_node.y += self.expand_dis * math.sin(theta)
20:       new_node.parent = nind
21:       if not self.collision_check(new_node, self.path.obstacles):
22:           continue
23:       self.node_list.append(new_node)
24:       # Verifica si se llegó al destino.
25:       dx = new_node.x - self.end.x
26:       dy = new_node.y - self.end.y
27:       d = math.sqrt(dx**2 + dy**2)
28:       if d <= self.expand_dis:
29:           goal_flag = True
30:           print("Goal!!")
31:           break
```

En la línea 9, se inserta el nodo inicial en la estructura del árbol de nodos.

En la línea 10, se declara el límite de repeticiones del algoritmo. Este límite está dado por el valor que se le da a la constante MAX_ITER.

Entre las líneas 13 y 18 se realiza la búsqueda aleatoria del nodo dentro del espacio de configuraciones.

En la línea 22, se obtiene el vecino más próximo de

la configuración aleatoria obtenida anteriormente.
Entre las líneas 21 y 23, se determina el nuevo nodo a agregar al árbol realizando un salto (de tamaño self.expand_dis * math.cos(theta)) partiendo del vecino más cercano y el nodo aleatorio. Si se determina, que en el camino no hay una colisión, se agrega el nuevo nodo al árbol. Por el contrario, si efectivamente hay un obstáculo de por medio, se continúa con la iteración siguiente y no se agrega el nodo al árbol resultante.
Entre las líneas 24 y 31, se verifica si se llegó al punto destino. Si la distancia entre el nuevo nodo y el punto destino es menor a la métrica establecida (un centímetro), se termina el procesamiento, obteniendo el árbol de nodos. Si la distancia no es menor, se continúa con la iteración siguiente.

Se implementó una interfaz gráfica utilizando node-red-dashboard [24], que permite visualizar el resultado de la ejecución de la simulación del algoritmo RRT.
Se puede observar en la figura 10, el resultado de simular la ejecución del algoritmo RRT tomando como punto de inicio la ubicación del Robot Rover 4WD (punto verde), el punto de destino (punto azul) y los obstáculos de por medio (puntos rojos).

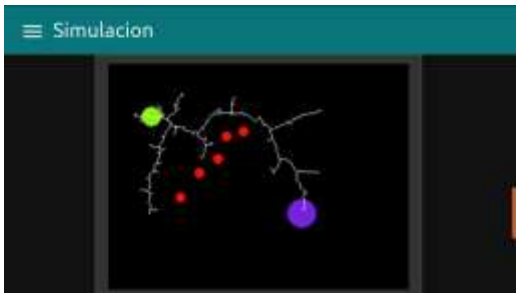


Fig. 10 Simulación de la ejecución del algoritmo RRT.

Como resultado de la planificación se obtiene un vector, donde cada una de sus celdas posee una coordenada (x,y). Se debe transformar estas coordenadas en movimientos. Cada movimiento constará de un grado de giro y una distancia en centímetros. Para dicha transformación se utilizar el algoritmo 5.

Algoritmo 5 Transformación a movimiento

```
1:  def load_movements(self, prev_deg):
2:      p1 = None
3:      for p2 in coords:
4:          if p1:
5:              distance = helper.get_distance(p1,
        p2, self.config.pixel_distance)
6:              xDiff = p2[0] - p1[0]
7:              yDiff = p2[1] - p1[1]
8:              degree = 360 -
        (math.degrees(math.atan2(yDiff,
        xDiff))%360)
9:              degree_difference = ((degree -
        prev_deg) + 180) % 360 - 180
10:             prev_deg = degree
11:   self.movements.append(degree_difference,
        distance])
12:         p1 = p2
```

En la Línea 1, se declara el encabezado de la función, que consta de un parámetro llamado "prev_degree". Este es el grado inicial del Robot Rover 4WD; a partir de este grado se podrá ir calculando la diferencia de grado entre los demás puntos.
En la línea 3, comienza el bucle, iterando por todas las coordenadas resultantes de la planificación del camino.
En la línea 4, se calcula la distancia Euclidiana ya mencionada anteriormente.
En las líneas 6 y 7, se calculan las diferencias entre x2-x1, y2-y1.
En las líneas 8 y 9, se calculan la diferencia de grados entre el punto anterior y un nuevo punto. Los grados varían en 0, 180 y 0, -180.
Por último, en la línea 12 se almacena el movimiento.

4.3. Plataforma WEB

Se desarrolló una plataforma WEB que brinda una interfaz gráfica; esta permite realizar la calibración de las cámaras como así también, visualizar el resultado de la ejecución del algoritmo de planificación de caminos RRT simulado. Por otro lado, permite dar inicio al sistema de multi-robots, visualizando en tiempo real todas las acciones que se estén llevando a cabo con el Robot Rover 4WD, los resultados de la planificación de la trayectoria gráficamente y las imágenes capturadas, tanto del vehículo autónoma como de la cámara aérea.

4.3.1. Calibración

Esta sección de la plataforma WEB, permite al usuario el ingreso de la información necesaria para calibrar las cámaras, como ser: tipo de figura geométrica ("cuadrado" o "circulo"), medida de ancho de la figura, etc.. Además, permite efectuar la calibración tanto del Robot Rover 4WD y la cámara aérea (Drone) notificando con un mensaje de éxito o de error.

Fig. 11 Interfaz de Calibración.

4.3.2. Simulación

Esta sección de la plataforma WEB, permite al usuario simular la planificación de un camino, permitiendo ingresar un punto de inicio, un punto de destino y diversos puntos de obstáculos, dando a elegir el radio y color de los mismos, junto con opciones para eliminar los obstáculos y para resetear la simulación. Además, permite la posibilidad de visualizar el resultado final o la animación de cómo el algoritmo de planificación RRT fue creando el árbol de nodos.

Fig. 12 Interfaz de Simulación sin animación.

Fig. 13 Interfaz de Simulación con animación.

4.3.3. Tiempo Real (Live)

Esta última sección de la plataforma WEB, permite dar inicio y fin al sistema Multi-Robots, y visualizar en tiempo real qué información se está intercambiando y en qué situación está cada actuador. Se puede apreciar gráficamente el resultado de la planificación del camino, y ver las imágenes que las cámaras están enviando a la instancia EC2 en el Cloud

Fig. 14 Interfaz de gestión del sistema Multi-Robots.

5. Resultados obtenidos

En esta investigación se ha implementado un sistema multi-robot conectado al Cloud público de AWS. Se han confeccionado los prototipos para emular un vehículo autónomo de 4 ruedas y un dispositivo de vuelo ambos no tripulados. Se han desarrollado los algoritmos que permiten realizar procesamiento de imágenes, reconocimiento de figuras geométricas, determinación del camino óptimo.

Además, se implementó una plataforma WEB para poder gestionar el sistema multi-robot como así también, poder visualizar el resultado de simular la ejecución del algoritmo de planificación de camino óptimo RRT.

Uno de los avances más significativos de esta investigación consiste en que, actualmente, el vehículo de cuatro ruedas conectado a AWS, se puede orientar, desplazar y arribar al destino establecido, de forma autónoma.

Podemos concluir y reafirmar que Cloud Robotics es una tecnología que favorecerá el avance de los sistemas multi-robots reduciendo las limitaciones que se presentan actualmente.

Referencias

[1] A. De Giusti, I. Rodríguez, M. Costanzo and M. Boggia, “Cloud Robotics: Auto Rover 4WD y Cuadricóptero controlados remotamente desde AWS” in CACIC ’17. *Proceedings of XVIII Workshop de Procesamiento Distribuido y Paralelo (WPDP) – XXIII Congreso Argentino de Ciencias de la Computación. La Plata, Argentina*. 2017.

[2] P. Mell and T. Grance, “The NIST Definition of Cloud Computing”. Publicación Especial 800-145. Septiembre, 2011.

[3] J.E. Pettoruti, I. Rodriguez, F. Chichizola, A. De Giusti. “Análisis de la degradación de las comunicaciones en algoritmos de cómputo científico en un Cloud privado”. In CACIC ’12. *Proceedings of XII Workshop de Procesamiento Distribuido y Paralelo (WPDP) – XVIII*

Congreso Argentino de Ciencias de la Computación. Bahía Blanca, Argentina, 2012.

[4] I. Rodriguez, J.E. Pettoruti, F. Chichizola and A. De Giusti: "Despliegue de un Cloud Privado para entornos de cómputo científico". In CACIC '11. *Proceedings of XI Workshop de Procesamiento Distribuido y Paralelo (WPDP) - XVII Congreso Argentino de Ciencias de la Computación. La Plata, Argentina*. 2011.

[5] "RoboEarth" Available at: http://www.roboearth.org. Accessed on 2018-06-06.

[6] J. Kuffner. "Cloud-enabled robots". *IEEE-RAS International Conference on Humanoid Robot. Nashville, USA*, 2010.

[7] L. Wang, M. Liu, M. Meng, R. Siegwart. "Towards Real-Time Multi-Sensor Information Retrieval in Cloud Robotic System". *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI). Hamburgo, Alemania*, 2012.

[8] L. Turnbull: "Cloud Robotics: Formation Control of a Multi Robot System Utilizing Cloud Infrastructure". *Proceedings of IEEE – Southeastcon. Jacksonville, USA*, 2013.

[9] "AWS IoT" Available at: https://aws.amazon.com/es/iot. Accessed on 2018-06-06.

[10] I. Bermudez, S. Traverso, M. Mellia and M. Munafò. "Exploring the cloud from passive measurements: The Amazon AWS case". *Proceedings of IEEE – INFOCOM*, 2013.

[11] L. Atzori, A. Iera and G. Morabito. "The Internet of Things: A survey". *Journal Computer Networks ELSEVIER*. Volume 54, Issue 15. 2010.

[12] "Protocolo MQTT" Available at: http://mqtt.org. Accessed on 2018-06-06.

[13] "OASIS MQTT", Available at: https://www.oasis-open.org. Accessed on 2018-06-06.

[14] "AWS IoT" Available at: https://aws.amazon.com/es/iot/. Accessed on 2018-06-06.

[15] "Node-RED" Available at: https://nodered.org. Accessed on 2018-06-06.

[16] "The official site RapsberryPi" Available at: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/. Accessed on 2018-06-06.

[17] "Raspirobot Board V3" Available at: https://www.monkmakes.com/rrb3/. Accessed on 2018-06-06.

[18] "Tipos de instancias de Amazon EC2" Available at: https://aws.amazon.com/es/ec2/instance-types/. Accessed on 2018-06-06.

[19] "Open Source Computer Vision Library" Available at: https://opencv.org/. Accessed 2018-06-06.

[20] "Python Software Fundation" Available at: https://www.python.org/. Accessed on 2018-06-06.

[21] J.J. Esqueda Elizondo and L.E. Palafox Maestre, *Fundamentos de procesamiento de imágenes*. Baja California, México: Universidad Autónoma de Baja California, 2005.

[22] A. Hernandez Zavala and J.A. Huerta Ruelas, "Sistema de medición de distancia mediante imágenes para determinar la posición de una esfera utilizando el sensor Kinect XBOX", *Journal of Polibits*, vol 49, 2014.

[23] D.A. López García, *Nuevas aportaciones en algoritmos de planificación para la ejecución de maniobras en robots autónomos no holonomos*. PhD thesis, Universidad de Huelva, 2011.

[24] "Node-Red-Dashboard" Available at: https://github.com/node-red/node-red-dashboard. Accessed on 2018-06-06.

SMOTE-BD: An Exact and Scalable Oversampling Method for Imbalanced Classification in Big Data

María José Basgall[1,2], Waldo Hasperué[2], Marcelo Naiouf[2], Alberto Fernández[3], and Francisco Herrera[3]

[1]*UNLP, CONICET, III-LIDI, La Plata, Argentina*
[2]*Instituto de Investigación en Informática (III-LIDI), Facultad de Informática - Universidad Nacional de La Plata*
[3]*DaSCI Andalusian Institute of Data Science and Computational Intelligence, University of Granada, Granada, 18071, Spain*
{mjbasgall, whasperue, mnaiouf}@lidi.info.unlp.edu.ar
{alberto, herrera}@decsai.ugr.es

Abstract

The volume of data in today's applications has meant a change in the way Machine Learning issues are addressed. Indeed, the Big Data scenario involves scalability constraints that can only be achieved through intelligent model design and the use of distributed technologies. In this context, solutions based on the Spark platform have established themselves as a de facto standard.

In this contribution, we focus on a very important framework within Big Data Analytics, namely classification with imbalanced datasets. The main characteristic of this problem is that one of the classes is underrepresented, and therefore it is usually more complex to find a model that identifies it correctly. For this reason, it is common to apply preprocessing techniques such as oversampling to balance the distribution of examples in classes.

In this work we present SMOTE-BD, fully scalable preprocessing approach for imbalanced classification in Big Data. It is based on one of the most widespread preprocessing solutions for imbalanced classification, namely the SMOTE algorithm, which creates new synthetic instances according to the neighborhood of each example of the minority class. Our novel development is made to be independent of the number of partitions or processes created to achieve a higher degree of efficiency. Experiments conducted on different standard and Big Data datasets show the quality of the proposed design and implementation.

Keywords: Big Data, Imbalanced classification, Preprocessing, SMOTE, Spark

1 Introduction

In Machine Learning, imbalanced data classification occurs when the classes in a problem show a skewed distribution [1, 2]. Canonical classifiers are designed to optimize overall accuracy not taking into account the relative class distribution. Hence, these classifiers tend to ignore small classes while concentrating on classifying the large ones accurately. This topic is very significant due to the large amount of real applications, where the minority class represents the key concept (e.g., many biological problems) [3].

A strategy to deal with imbalanced datasets consists of applying a preprocessing step to resampling the training data. To do so, the most popular technique is known as SMOTE ("Synthetic Minority Oversampling TEchnique") [4, 5], which forms new minority class examples by interpolating between several neighbour minority class examples.

Preprocessing methods were initially designed for standard size datasets [6]. As such, they cannot be applied directly when it comes to Big Data, due to scalability issues [7]. To overcome this, those techniques must be adapted and/or reimplemented to be executed in a distributed way, using frameworks such as Apache Spark [8, 9].

However, the translation from the original preprocessing method towards a distributed approach is not straightforward. First, the programming framework may imply a complete redesign of the procedure to be adapted to a divide-and-conquer strategy, e.g., MapReduce. Second, the data division required to address Big Data problems may lead to lack of data when processing local models. For these reasons, there is still few research on the topic [10].

These facts imply the need to develop a new research line on the generation of minority instances for Big Data. It is important to point out that current technologies to work with Big Data present two different approaches with respect to how partial models from the Map stage are aggregated [11]. On the one hand, there are "local" strategies which produce an approximate model by applying a direct aggregation on partial models. On the other hand, there are "global" methods, which distribute both data and models to iteratively build a final result. It is straightforward to notice that the first type of approaches are preferred when seeking for an exact solution.

In this work, we propose SMOTE-BD, an exact solution for the implementation of SMOTE in Big Data. To do so, our methodology focuses on the calculation of the neighborhood based on a recent k nearest neighbors (kNN) approach [12]. Additionally, it incorporates a thorough design based on the use of scalable data structures and functions. Specifically, it is implemented under Scala for the Spark framework [9].

A brief experimental study is carried out in order to show the quality of the proposed oversampling implementation. First, a comparison between SMOTE-BD and the standard sequential methodology is shown for small datasets. Then, we contrast the algorithm's performance and scalability in the scenario of Big datasets using different numbers of partitions.

The remainder of this paper is organized as follows. Section 2 introduces the current state of the art in imbalanced Big Data classification. Section 3 details the proposed model of a fully scalable SMOTE in Spark framework. Then, in Section 4 the experimental study carried out. Finally, in Section 5 the conclusions and future works are described.

2 Imbalanced Big Data Classification

The problem of imbalanced classification appeared at the same time that researchers realized that traditional classification algorithms could not model well the underrepresented classes [1, 2]. When it comes to the Big Data context, this problem is accentuated.

In [10], authors presented an exhaustive study with the objective of evaluating the performance of the traditional solutions for class imbalanced in the Big Data context. To this end, several preprocessing techniques were adapted and embedded in a MapReduce workflow. Specifically, in this research the random oversampling (ROS-BigData), random undersampling (RUS-BigData) and SMOTE MapReduce version for Hadoop (SMOTE-H) were employed.

Following the Hadoop MapReduce philosophy, each Map process was responsible for adjusting the class distribution for its data partition, either by random replication of minority class instances (ROS-BigData), random elimination of majority class instances (RUS-BigData) or the generation of synthetic data carried out by SMOTE technique (SMOTE-H). Subsequently, a single Reduce process was responsible for collecting the results generated by each mapper and assigning them randomly to form the balanced dataset.

All those preprocessing methods worked locally within each Map, thus limiting the potential of these algorithms. As remarks of the aforementioned work, they observed random oversampling to be more robust than the other techniques when the number of data partitions increased.

The MapReduce solutions, which are based on the "divide-and-conquer" approach implies a partitioning of the data that can lead to two serious consequences [1]. One of them is the extreme lack of positive data in each partition data, which represents a partial representation of the dataset information, particularly with regard to the real neighborhood of the instances. The other consequence of the partitioning, and very related to the previous one, is the presence of very local data with low density called "small-disjuncts" which represents the concepts of interest. Creating minority instances from small-disjuncts can lead to noise or an over-generalization, and they would enter the "safe" zones of the majority class.

Figure 1 depicts the lack of data situation for the training data of the *yeast5* imbalanced problem from KEEL dataset repository [13]. It can be noticed the low concentration of minority instances that they can be considered as noise or rare data.

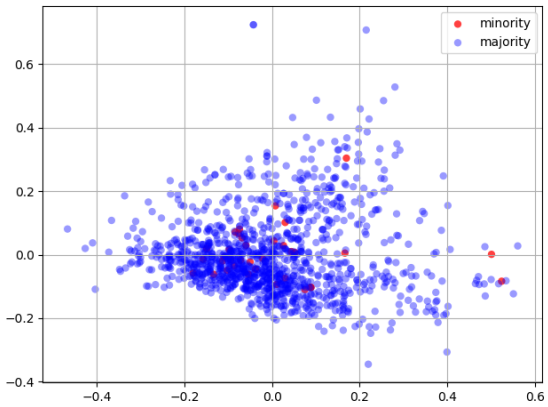


Figure 1: Lack of density on the *yeast5* dataset

3 SMOTE-BD: An exact oversampling solution in Spark

In this section, an exact SMOTE fully scalable methodology in Spark for Big Data is presented.

Figure 2 depicts the general scheme of this proposal. First, the algorithm performs a filtering over the training set (stored in the HDFS) to get the minority and majority subsets of instances. Then, the minority data, which is partitioned according to an algorithm parameter, is normalized taking into account the statistics of the full training set and is cached to be reused in the following steps.

Later, nearest neighbors for each positive instance is obtained using an exact implementation of kNN in Spark (kNN-IS) [12] which splits the training dataset in a user-defined number of partitions, calculates for each instances in a chunk its neighbors and finally, in a reduction phase, makes a final list of k nearest neighbors.

After that, the generation of artificial minority class instances is begun. All the nearest neighbors obtained in the previous step are broadcasted to the main memory of the all nodes in the cluster. The *broadcast*

operation allows to keep a read-only variable cached on each node rather than shipping a copy to each task, and it performs this action in an efficient manner.

Then, for each positive instance in a data partition and using the broadcasted variable, the algorithm generates the corresponding number of synthetic examples by interpolating between each minority instance and its k nearest neighbors. Figure 3 depict how to create synthetic data points in the SMOTE algorithm.

Finally, the algorithm performs a denormalization process over the artificial dataset and joins the original positive and negative instances with the artificial ones in order to conform the balanced dataset, and saves it in the HDFS.

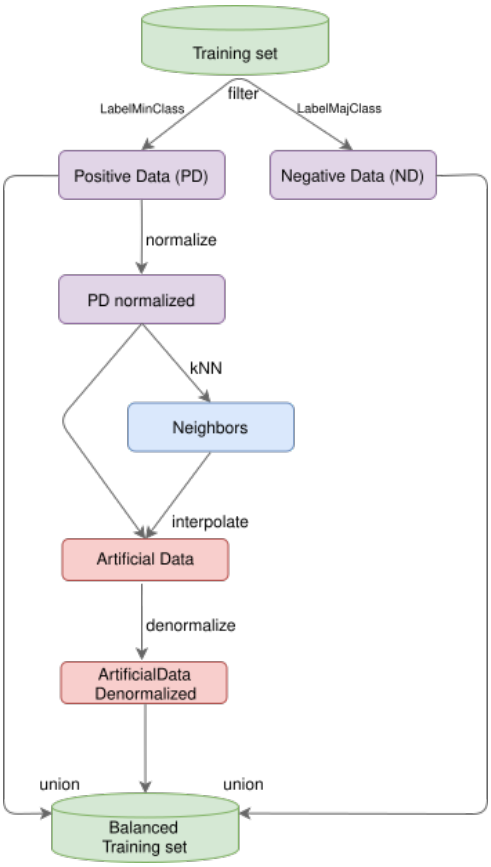


Figure 2: SMOTE-BD flowchart

Algorithms 1 and 2 show a pseudocode of the sequence of actions described above . The former covers the main program and the latter the function to create each artificial instance. This function invokes another function called *interpolation* which is in charge of doing the interpolation between two points. There is no pseudocode of this due to its simplicity.

4 Experimental Study: Analysis of the behavior of SMOTE-BD

In this section, the performance achieved by classification algorithms in synergy with SMOTE-BD implementation is presented. In order to compare the

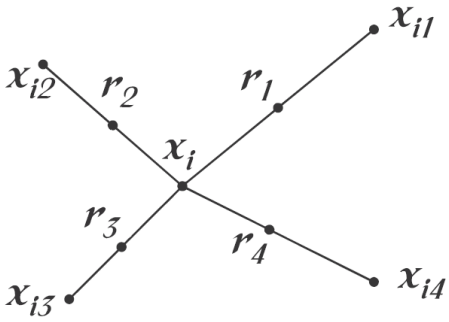


Figure 3: Interpolation between a minority instance and its k nearest neighbors.

performance of SMOTE, eight imbalanced datasets were selected and divided into two categories based on the number of examples which each dataset contains. Table 1 shows the datasets summary, where the number of examples (#Ex.), number of attributes (#Atts.), class name of each class (majority and minority), number of instances for each class, class distribution and imbalance ratio (IR) are included.

The behavior of the resultant preprocessed datasets was tested using the Decision Trees classifier (DT), implemented in the Spark's MLlib library [14]. Table 2 shows the parameters used for the methods according to their authors' specification.

Regarding the infraestructure used to perform the experiments, the Hadoop cluster at University of Granada was used, which consists of fourteen nodes connected via a Gigabit Ethernet network. Each node has a Intel Core i7-4930K microprocessor at 3.40GHz, 6 cores (12 threads) and 64 GB of main memory working under Linux CentOS 6.9. The cluster works with Hadoop 2.6.0 (Cloudera CDH5.8.0), where the head node is configured as NameNode and ResourceManager, and the rest are DataNodes and NodeManagers. Moreover, the cluster is configured with Spark 2.2.0.

The quality measures of classification are built from a confusion matrix (shown in Table 3), which organizes the samples of each class according to their correct or incorrect identification. From this matrix four metrics that describe both classes independently are obtained:

True Positive Rate $TPR = \frac{TP}{TP+FN}$ is the percentage of positive instances correctly classified.

True Negative Rate $TNR = \frac{TN}{FP+TN}$ is the percentage of negative instances correctly classified.

False Positive Rate $FPR = \frac{FP}{FP+TN}$ is the percentage of negative instances misclassified.

False Negative Rate $FNR = \frac{FN}{TP+FN}$ is the percentage of positive instances misclassified.

None of these rates alone are adequate independently, therefore more robust metrics exist to evaluate

Algorithm 1 SMOTE-BD algorithm

Require: Tr, Ts, ratio, k, nP, nR, nIt, minClassLabel

$origData \leftarrow textFile(Tr)$
$minData \leftarrow origData.filter(labels == minClassLabel)$
$minData \leftarrow minData.map(normalize).repartition(nP)$
$neighbors \leftarrow kNN_IS.setup(Tr,Tr,k,nR,nI).calculatekNeighbours()$
$crFactor \leftarrow (nMaj - nMin)/nMin$
$neighbors \leftarrow broadcast(neighbors)$
$balancedData = null$
$synData = null$
for $i < nIt$ **do**
    $synTmp \leftarrow minData.mapPartitionsWithIndex(createSynthData(index, partData, neighbors, crFactor, k))$
    if $synData == null$ **then**
        $synData \leftarrow synTmp$
    else
        $synData \leftarrow synData.union(synTmp)$
    end if
end for
$synData \leftarrow synData.map(denormalize)$
$balancedData \leftarrow synData.union(origData)$

Algorithm 2 Function to create synthetic instances between the minority class examples and their neighbors

procedure CREATESYNTHDATA($index, partData, neighbors, crFactor, k$)
    $artificialData = null$
    for $firstInstance \leftarrow partitionData; nc = 0$ *to* $crFactor$ **do**
        $selNeighbor \leftarrow newRandom().nextInt(k)$
        $secondInstance \leftarrow neighbors(selNeighbor)$
        $newIntance \leftarrow interpolation(firstInstance, secondInstance)$
        $artificialData.add(newInstance)$
    end for
    return $artificialData$
end procedure

Table 1: Datasets summary

Big datasets	#Ex.	#Atts.	Class (maj;min)	#Class(maj; min)	%Class(maj; min)	IR
covtype7	464677	54	(negative; positive)	(448421; 16256)	(96.5; 3.5)	27.58
poker0_vs_2	450022	10	(negative; positive)	(410960; 39062)	(91.32; 8.68)	10.52
poker0_vs_5	412600	10	(negative; positive)	(410960; 1640)	(99.60; 0.4)	250.58
susy_ir4	2712175	18	(negative; positive)	(2169740; 542435)	(80; 20)	4
Small datasets	**#Ex.**	**#Atts.**	**Class (maj;min)**	**#Class(maj; min)**	**%Class(maj; min)**	**IR**
page-blocks0	5472	10	(negative; positive)	(4913; 559)	(89.78; 10.22)	8.78
segment0	2308	19	(negative; positive)	(1979; 329)	(85.75; 14.25)	6.02
shuttle-c0-vs-c4	1829	9	(negative; positive)	(1706; 123)	(93.28; 6.72)	13.88
yeast5	1484	8	(negative; positive)	(1440; 44)	(97.04; 2.96)	32.78

performance in classification scenarios. One of the most widely used metric in imbalanced classification is called Geometric Mean (GM) [15] which is defined in equation 1. The GM attempts to maximize the accuracy of each one of the two classes at the same time.

$$GM = \sqrt{TPR * TNR} \quad (1)$$

The results to apply SMOTE sequential implementation (available in KEEL Software Tool [13]) and SMOTE-BD on small datasets are shown in Table 4 and depicted in Figure 4. For SMOTE-BD implementation, tests with different number of data partitions (1 to 4) have been performed.

Table 5 shows the average GM results in training and testing sets for the Spark SMOTE implementation using 1, 8 and 32 partitions over the four Big Data cases of study.

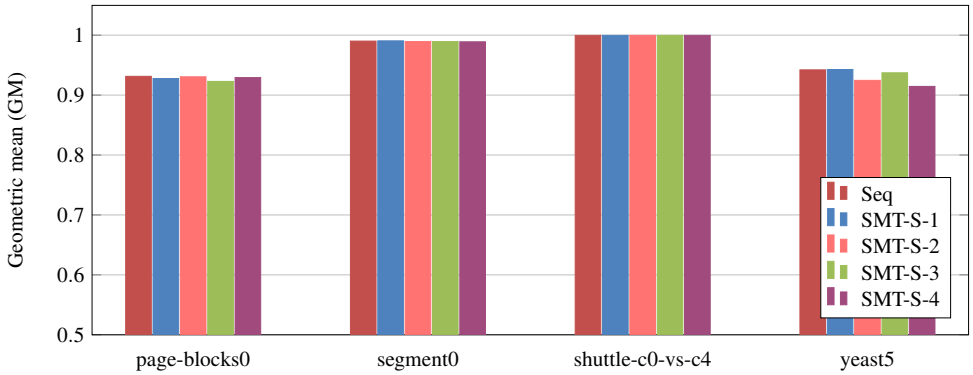


Figure 4: Average results of the sequential SMOTE and the SMOTE-BD version, combined with the Spark version of Decision Trees over the small datasets using the GM measure.

Table 2: Algorithms and parameters

Algorithms	**Parameters**
SMOTE Seq	k Nearest Neighbors: 5 Distance function: Euclidean Desired IR: 1
SMOTE-BigData	k Nearest Neighbors: 5 Distance function: Euclidean Desired IR: 1 Number of partitions: 1 Number of reducers: 1

Table 3: Confusion matrix for performance evaluation of a binary classification problem

Actual	Predicted	
	Positive	Negative
Positive	True positive (TP)	False negative (FN)
Negative	False positive (FP)	True negative (TN)

5 Conclusions and future works

In this work, we have developed SMOTE-BD, a fully scalable oversampling technique for imbalanced classification in Big Data Analytics. Two main reasons have motivated the design of this new methodology. On the one hand, the lack of current solutions for such a significant area of study. On the other hand, to consider a global procedure that takes into account the whole neighborhood of each minority class instance.

The advantages of this novel approach are clear. The most significant one is consolidating a cluster of new synthetic instances, thus avoiding the overgeneralization due to data locality for traditional MapReduce procedures. Furthermore, the number of data partitions considered to add more efficiency to the process is independent of the algorithm's procedure, so that there are no constraints regarding the lack of data.

The novelty of this area of research implies many topics for future study. Among them, we may stress three important points. First, to consider novel data structures to avoid the limitation of "broadcast" variables for very large dataset sizes. Second, to analyze the quality of the new preprocessed data regarding different parametrization, especially the percentage of oversampling. Finally, to study new scalable designs of SMOTE-based algorithms in Big Data, in particular considering the data redundancy that is present in current datasets.

References

[1] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Information Sciences*, vol. 250, no. 20, pp. 113–141, 2013.

[2] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 2016.

[3] D. Galpert, A. Fernández, F. Herrera, A. Antunes, R. Molina-Ruiz, and G. AgÃŒero-Chapin, "Surveying alignment-free features for ortholog detection in related yeast proteomes by using supervised big data classifiers," *BMC Bioinformatics*, vol. 19, no. 1, 2018.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over–sampling technique," *Journal of Artificial Intelligent Research*, vol. 16, pp. 321–357, 2002.

[5] A. Fernandez, S. Garcia, F. Herrera, and N. Chawla, "Smote for learning from imbalanced data: Progress and challenges. marking the 15-year anniversary," *Journal of artificial intelligence research*, vol. 61, pp. 863–905, 2018.

Table 4: Average results of the sequential SMOTE and the SMOTE-BD version, combined with the Spark version of Decision Trees over the small datasets using the GM measure.

Datasets	Seq	SMT-1p	SMT-2p	SMT-3p	SMT-4p
page-blocks0	0.9313	0.9276	0.9305	0.9228	0.9292
segment0	0.9901	0.9905	0.9893	0.9893	0.9889
shuttle-c0-vs-c4	0.9997	0.9997	0.9997	0.9997	0.9997
yeast5	0.9421	0.9427	0.9246	0.9372	0.9145

Table 5: Average results of the SMOTE-BD algorithm for 1, 8 and 32 partitions, combined with the Spark version of Decision Trees over the big datasets using the GM measure.

Datasets	SMT-1p	SMT-8p	SMT-32p
covtype7	0.9227	0.9287	0.9181
poker0_vs_2	0.4582	0.4585	0.4713
poker0_vs_5	0.4162	0.4208	0.4206
susy_ir4	0.7615	0.7601	0.7585

[6] R. C. Prati, G. E. A. P. A. Batista, and D. F. Silva, "Class imbalance revisited: a new experimental setup to assess the performance of treatment methods," *Knowledge and Information Systems*, vol. 45, no. 1, pp. 247–270, 2015.

[7] C. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on Big Data," *Information Sciences*, vol. 275, pp. 314–347, 2014.

[8] A. Fernández, S. Río, V. López, A. Bawakid, M. J. del Jesus, J. Benítez, and F. Herrera, "Big Data with cloud computing: An insight on the computing environment, MapReduce and programming framework," *WIREs Data Mining and Knowledge Discovery*, vol. 4, no. 5, pp. 380–409, 2014.

[9] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *HotCloud 2010*, pp. 1–7, 2010.

[10] A. Fernandez, S. del Rio, N. V. Chawla, and F. Herrera, "An insight into imbalanced big data classification: Outcomes and challenges," *Complex and Intelligent Systems*, vol. 3, no. 2, pp. 105–120, 2017.

[11] S. Ramírez-Gallego, A. Fernández, S. García, M. Chen, and F. Herrera, "Big data: Tutorial and guidelines on information and process fusion for analytics algorithms with mapreduce," *Information Fusion*, vol. 42, pp. 51–61, 2018.

[12] J. Maillo, S. Ramírez-Gallego, I. Triguero, and F. Herrera, "knn–is: An iterative spark-based design of the k-nearest neighbors classifier for big data.," *Knowledge-Based Systems*, vol. 117, pp. 3–15, 2017.

[13] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "KEEL data–mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *Journal of Multi–Valued Logic and Soft Computing*, vol. 17, no. 2-3, pp. 255–287, 2011.

[14] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. J. Franklin, R. Zadeh, M. Zaharia, and A. Talwalkar, "MLlib: Machine learning in apache spark," *Journal of Machine Learning Research*, vol. 17, no. 34, pp. 1–7, 2016.

[15] R. Barandela, J. S. Sánchez, V. García, and E. Rangel, "Strategies for learning in class imbalance problems.," *Pattern Recognition*, vol. 36, no. 3, pp. 849–851, 2003.

Cloud Médicos - Privacidad, integridad y estandarización – Estado actual

Ezequiel Velurtas[1], Patricia Bazán[2]

evelurtas@yahoo.com.ar , pbaz@info.unlp.edu.ar

[1] - *Facultad de Informática – UNLP, 2 – LINTI – Facultad de Informática - UNLP*

Resumen

La presente investigación, plantea un análisis comparativo y estado del arte de los servicios de almacenamiento de datos y métricas biomédicas – conocidos como *cloud* médicos – en sus versiones privativas y abiertas. Entre las variantes privativas de analizarán *cloud* médicos globales privados como Samsung *SHealth*, *Apple Healthcare*, *DrIcloud*, *ClinicCloud* y entre las abiertas, *OpenEHR*, registro de salud electrónicos, Receta Única Electrónica España y SIISA (Sistema Integrado de Información Sanitaria Argentina).

Palabras Claves

Cloud, Integridad de datos, Seguridad, Estandarización.

1. Introducción

El crecimiento y demanda actuales de las tecnologías *wearables* - dispositivos o conjunto de dispositivos que pueden portarse, e interactúan con el usuario y otros dispositivos -, utilizados para relevamiento biomédico personal, se representan mediante bandas o relojes inteligentes con sensores biométricos que brindan información de utilidad para entrenamiento deportivo y también en el terreno de la salud.

Si bien el mercado está poniendo el foco en este nicho tecnológico, la diversidad de modelos y marcas, sumado a la ausencia de estándares y protocolos comunes, lleva a un desaprovechamiento de los beneficios y recursos y aumenta complejidad a su adopción.

En el 2017 como se indica en el reciente informe del IDC - *International Data Corporation*, principal proveedor mundial de inteligencia de mercado, servicios de asesoramiento y eventos para los mercados de tecnologías de la información, telecomunicaciones y tecnología de consumo [10] - se vendieron 61,5 millones de unidades y se espera 149,5 millones para el 2021. Todas ellas unidades *SmartWatches* y *SmartBands* de la familia *Android Wear*[*3] - producto de código abierto que empieza a marcar un hilo conductor - y también las pertenecientes a *Apple* con su *Apple Watch* and *Health*,

Bajo este escenario entran en juego los servicios de almacenamiento biomédicos y métricas conocidos como *Cloud* Médicos los cuales son los repositorios donde se va almacenando cada uno de estos datos. Esta información si bien es de tamaño pequeño (una georreferencia o un valor de medición biomédico), por su continuidad y segmentación genera gran volumen, por lo cual estos *cloud* médicos terminan siendo también grandes almacenes de *Big Data* [2] con la complejidad que eso refiere. Hoy los usuarios están cautivos de la marca del dispositivo que se utilice y resulta poco claro el uso de los datos recolectados, siendo útiles en función de donde se alojen los mismos para darle a las trazas un uso médico legítimo y avalado.

La consolidación de los datos recolectados por estos dispositivos y la generación de umbrales personalizados y parametrizables aclararían el mercado y darían un nuevo foco de negocios, captando mayor cantidad de adeptos a esta tecnología y la enfocaría al cuidado y prevención médica de manera mucho más confiable.

/

[*1] **Android Wear o Wear OS** : es el sistema operativo para dispositivos corporales basado en Android que Google presentó a la sociedad el 18 de marzo de 2014.

[*2] **Big Data**: Nos referimos a conjuntos de datos o combinaciones de conjuntos de datos cuyo tamaño , complejidad y velocidad de crecimiento dificultan su captura, gestión, procesamiento o análisis mediante tecnologías y herramientas convencionales,

[*3]**OpenEHR**: es una comunidad virtual de trabajo con el objetivo de convertir los datos de salud de la forma física en forma electrónica y garantizar la interoperabilidad universal

entre todas las formas de datos electrónicos. El enfoque principal de su esfuerzo está en los registros

2. Metodología

La investigación fue de carácter descriptivo, se llevó a cabo analizando las principales publicaciones, las tendencias del mercado, la influencia de las grandes empresas productoras de wereables y luego comparándolos con los casos de estudio de mayor relevancia.
La Tabla 1 enumera los trabajos seleccionados para su análisis para los cuales se muestran las palabras claves consideradas y los principales temas que abarcan.

Tabla 1 - Principales Publicaciones

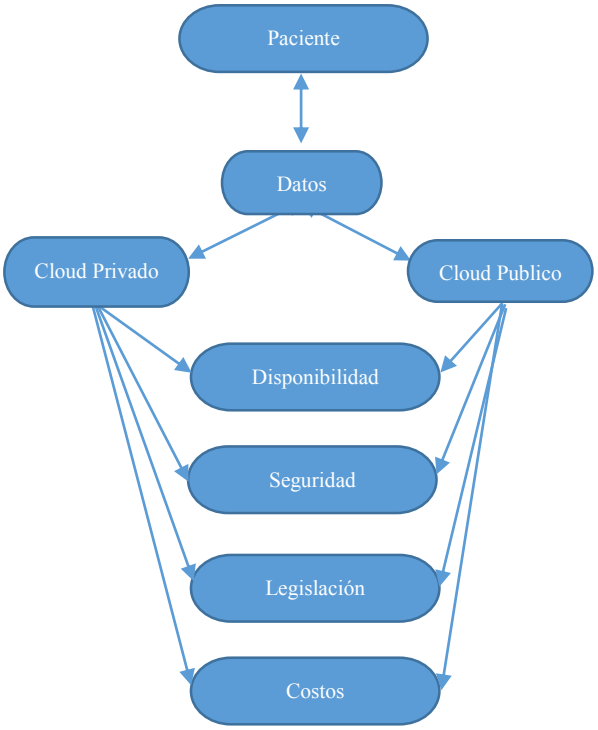
Título del Trabajo	Referencias	*Temas Claves*	Palabras claves
Cloud Computing in Health	*2012, Canada Health Info way Inc*	*Porque Nube para la salud, seguridad y privacidad de los datos*	*ehealth, Cloud Publico*
Social Innovation in Healthcare	*2016, Frost& Sullivan*	*Convergencia de Salud, innovacion en Salud*	*Tendencias , EHR , Costos*
Impact of Cloud Computing onHealthcare V. 2.0	*2017, Cloud Standards Customer Council*	*Construccion de Modelos de Cloud Medicos y guia practica de estandares y puesta en Marcha de Cloud*	*Mercado , Estandares, Privacidad*

También se tomaron para la base de esta investigación, las regulaciones nacionales e internacionales sobre el tema de estudio, informes y presentaciones actuales sobre *cloud* médicos privados y federales, además de encuestas y estadísticas de IDC.
Se definieron las siguientes preguntas de investigación para el análisis y conceptualización de los trabajos elegidos:

- *¿Hay una necesidad de contar con cloud médicos?*
- *¿Deben los cloud médicos ser privados o federales?*
- *¿Existe Legislación que ampare o proteja los datos médicos en los cloud?*
- *¿Cuáles son las tendencias del mercado en cuanto al uso de Cloud médicos?*
- *¿Qué tipo de experiencias de uso de cloud médicos hay en el mundo ?*
- *¿Cuáles son las pautas para el crecimiento de los cloud médicos?*
- *Los cloud médicos, ¿son verdaderamente útiles para los pacientes?*

La Figura 1 muestra un mapa conceptual construido en base a las preguntas planteadas. Los tipos de *cloud* influyen en la manera en que se administran los datos y también sobre la utilidad para el paciente. integral de los ejes fundamentales del trabajo., el cual podemos ver en la Figura 1.

Imagen 1 Mapa conceptual.



Con la información procesada y analizada la investigación se plasmara en las siguientes secciones, Estado actual, Beneficios y Tendencias, Regulaciones, Desafíos a Enfrentar y se presentaran las conclusiones a las que se arribaron.

3. Estado Actual

En la actualidad los *cloud* médicos están siendo liderados por los de órbita privada, los cuales están imponiendo sus propios métodos de recolección y almacenamiento de los datos, haciendo cautivos a sus usuarios, quienes en muchos casos desconocen esta situación y adhieren a estos programas mediante

el uso de *wearable´s*, dejando sus datos personales biomédicos a merced y uso de empresas privadas, no conociendo la ubicación de sus datos, su uso y declaraciones de privacidad sobre los mismos.
En respuesta a estos problemas se está impulsando un programa abierto llamado OpenEHR*[3], al cual ya están adhiriendo varios países de manera activa como Países Bajos, Australia, Brasil, Noruega, Portugal, Rusia, Eslovenia, Suecia y Reino Unido.
Por otra parte desde las universidades también se está abordando esta temática mediante el análisis y aporte al modelado de la plataforma y *cloud* del programa aportando también, con casos de estudio y métricas. Este es el caso de países como Argentina, Brasil, Alemania, Japón, Nueva Zelanda, Portugal, España, Suecia y Reino Unido.
En el caso de España en particular si bien no es activo en el programa, se basó en OpenEHR e implementó la Receta Electrónica Única, la cual es una historia clínica común, tanto para la medicina privada como la estatal que permite un control y optimización de los gastos económicos en estudios, prácticas y venta de fármacos. El Estado es el administrador del *cloud* médico.
En este sentido, países de Europa y América del Norte están implementando sus *cloud* médicos federales lo cual otorga las siguientes ventajas: 1 - pacientes con libre elección de médico e institución, 2 - control absoluto de sus datos médicos, 3 - reducción de gastos en prácticas y estudios y 4 - Universalidad de los datos.
En Argentina la experiencia más cercana a un *cloud* medico formal ,son los sistemas de historias clínicas centrales que manejan casi todas las instituciones privadas , los cuales comparten la información de los pacientes solo con sus centros asistenciales repartidos en distintas locaciones; si bien la menara de almacenar los datos es en un cloud, dichos repositorios de información son de carácter privado de cada institución y no son accesibles por el paciente, u otras instituciones públicas o privadas, En la Tabla 2 podremos ver un resumen comparativo de las diferencias entre los cloud privados y los Federales :

Tabla 2. Comparación de cloud´s.

Características	**Cloud Privados**	**Cloud Federales**
Resguardo de Datos	*Por un Tercero*	*Por el Estado*
Autonomía del Paciente	*Cautivos de la institución.*	*Libre elección de prestador e institución*
Manejo de Datos	*No estandarizado*	*Estandarizado*
Gastos médicos	*Duplicados*	*Optimizados*
Recursos médicos	*Ineficienciente , Repetición de Estudios y practicas*	*Eficiencia*
Costo para el paciente	*Sin Regulación*	*Absorbido por el estado*
Legislación	*Dependiendo donde se hostee el servicio*	*Legislación marcada por el estado*

Si bien es posible encontrar algunas líneas de investigación en instituciones públicas, el impulso proviene fundamentalmente de las universidades nacionales que no siempre cuentan con los recursos suficientes para abarcar muchas instituciones de salud ni tampoco las más grandes, quedando los proyectos con alcance en focos específicos y sobre instituciones más pequeñas tomadas como casos de estudio.

Un ejemplo de lo mencionado es el programa de investigación PICIS (actualmente en pausa) que se basaba en la digitalización de las historias clínicas para todos los ciudadanos, del país [6] y era una de las metas del Plan Nacional de Informatización de la Salud (PICIS), lanzado en 2003. “La pausa en el proyecto se debe a que se deberían modificar normas vigentes que establecen que las prescripciones o recetas médicas deben ser manuscritas” (*Cita textual - Ex secretario de Relaciones Sanitarias del Ministerio de Salud, Eduardo Bustos Villar).*

Esta modificación de norma impidió que el proyecto pudiera tener avances desde hace varios años. El programa llegó a crear SIISA Sistema Integrado de Información Sanitaria Argentina, mediante el cual varias provincias Argentinas llegaron a registrar hasta el 2017, 19.000 establecimientos sanitarios, incluyendo hospitales , centros asistenciales y salas primarias de atención (sobre más de 80000 mil centros asistenciales existentes), más de 523.000 profesionales de la salud [10] *)* habilitados que registran diariamente toda la información de pacientes que reciben vacunas del Programa Nacional de Vacunación [11] y los medicamentos que se les suministra. La integración de SIISA con PICIS permitiría completar la registración de prácticas de laboratorio, imágenes y consultas profesionales.

En cuanto a *cloud´s* médicos privados más destacados como *DrIcloud*[*4], *ClinicCloud*[*5]. En el caso de los dos últimos son una solución en países que no tienen implementado un sistema público de *cloud* médico y permiten, según los acuerdos de confidencialidad con el paciente, mantener los datos lejos del Estado y de los sistemas nacionales, ya que aún no están regulados por ningún ente internacional.

En el caso de Samsung SHealth y Apple Healthcare, como se ha mencionado mantienen a sus usuarios cautivos de sus plataformas ya que al utilizar cualquiera de sus dispositivos con sonorización biométrica, las métricas son almacenadas y procesadas en sus *cloud* y puestas a disposición solo para sus aplicaciones móviles.

3.1 Beneficios y tendencias del *Cloud* Médico

El mercado de los *cloud* médicos y el e-Health (*Término con el que se define al conjunto de Tecnologías de la Información y la Comunicación que, a modo de herramientas, se emplean en el entorno sanitario en materia de prevención, diagnóstico, tratamiento, seguimiento, así como en la gestión de la salud, ahorrando costes al sistema sanitario y mejorando la eficacia de este),*está movido por los pacientes que pasan a ser consumidores y desempeñan un papel cada vez más importante en la toma de decisiones en el mercado de la salud, incidiendo ampliamente en la definición de planes de salud y tratamientos médicos. A esto se suma también que las personas tienen mayor conciencia sobre el cuidado de su salud y adoptan para ello el uso de *wereables* para registrar su actividad física, signos vitales y demás métricas biomédicas. Esta toma de conciencia, cada vez en más auge, empuja el mercado de la salud a transformarse y buscar soluciones basándose en las cinco tendencias más marcadas :[9]

1- *Escalada de consumo*: La alta demanda de para la realización estudios y prácticas médicas, ha llevado a los centros asistenciales a que optimicen sus costos de funcionamiento evitando gastos innecesarios y haciendo campañas de prevención como medida de ahorro.

2- *Impacto de la regulación de la salud y la reestructuración de los riesgos financieros*, Regulaciones tales como el Programa Internacional de Salud ONC Health TI Certificación [12] exige nuevas interfaces en los softwares de salud, incluyendo APIs y capacidades de acceso a datos. Estos requisitos están dando lugar a la necesidad de actualizar, mejorar o reemplazar los actuales sistemas de gestión médica.

3 -*Influencia de la digitalización*. La masificación de las bandas de entrenamiento (*fitness)* y las aplicaciones informáticas móviles permiten a los consumidores controlar sus métricas de salud y signos vitales. La Big Data recolectada sobre estos dispositivos ayuda a los consumidores a que sean más consientes sobre su salud y más capaces de tomar sus propias decisiones de atención preventiva.

4 -*Foco en la salud preventiva*. La inversión en prevención es un compromiso con los pacientes la cual reditúa en ahorro y una liberación de los centros asistenciales, para lo cual se apoyan en las aplicaciones móviles, el Internet de las cosas (IoT) y las tecnologías portátiles basadas en la nube.

5. *Transformación de la práctica médica*. Tanto en las instituciones como en los consultorios privados, este cambio viene de la mano de nuevas normas médicas y variación de las prácticas actuales que pueden variar significativamente la manera en que se practica la medicina. Dicha transformación se basa en análisis predictivo, basado en las biométricas de cada paciente y la planificación la atención.

Los beneficios más importantes obtenidos de la aplicación de estas nuevas tendencias, los podemos enfocar según cada destinatario en:

1- Profesionales de la salud

- Gestión completa de la prescripción médica
- Control completo sobre pacientes e historiales
- Se adapta a la filosofía de trabajo de cada médico, los cuales tienen disponible toda la historia clínica y estudios de los pacientes.
- Prescripción de tratamiento registrada
- Gestión automática de informes
- Registración sin papel

2-Para la gestión

- Permite una Gestión Global de los pacientes.
- Los procesos automatizados evitan errores humanos.
- Mejora la respuesta ante solicitudes de información.
- Optimiza costos en prácticas médicas y de gestión de las instituciones.
- Localización inmediata de las historias clínicas.
- Control y planificación de agendas y turnos.

3-Para los Pacientes

- Proactividad y prevención en su salud.
- Mejora la calidad de vida, desde la prevención y evita repetir estudios.
- Percepción de una gestión administrativa eficaz y un centro médico más moderno.
- Perciben un eficaz control de sus acontecimientos o incidentes medicos, .
- Incrementan su confianza.
- Incrementan sus creencias positivas hacia el profesional y las practicas realizadas.
- Permite realizar citas online.
- Administrar sus tiempos.

[4] **DrIcloud:** Software y Cloud Medico presente en varios países. Software para Clínicas y Consultorios en la Nube dedicado a expediente electrónico médico. https://dricloud.com/

[5] **ClinicCloud** Software médico para gestión de pacientes, para controlar analizar y administrar las historias del paciente en la nube.https://clinic-cloud.com/

[6] **SISA** Sistema Integrado de información Sanitario Argentino https://sisa.msal.gov.ar/sisa/ , https://www.argentina.gob.ar/sisa-mobile

[7] **Samsung SHealth** Cloud de datos y programa de Salud de la empresa Samsung

[8] **Apple Healthcare** Cloud de datos y programa de Salud de la empresa Apple.

[9] **IHS 2016** Actualización: Las complejidades del médico oferta y la demanda: Proyecciones 2014-2025 https://www.aamc.org/download/458082/data/2016_complexities_of_supply_and_demand_projections.pdf

[10] Estadísticas tomadas del sitio Oficial del ministerio de Salud

[11] https://www.argentina.gob.ar/salud/epidemiologia

12 **ONC Health TI :** Programa de certificación de TI de la Oficina del Coordinador Nacional de Tecnología de la Información (ONC) es un programa voluntario de certificación https://www.healthit.gov/policy-researchers-implementers/about-onc-health-it-certification-program

[13] **Dato Sensible:** Datos personales que revelan origen racial y étnico, opiniones políticas, convicciones religiosas, filosóficas o morales, afiliación sindical e información referente a la salud o a la vida sexual.

3.2 Regulaciones

Las regulaciones y legislaciones sobre los *cloud* médicos y el manejo de los datos sensibles de los pacientes fueron y son en algunos países un gris sin definir. La Unión Europa ha sido la precursora y recientemente puso en vigencia una nueva regulación la RGPD 679/2016 (previamente los proveedores de *cloud* debían ya cumplir con Directiva 95/46/CE/15/99 que fija los criterios de protección de datos y su libre circulación) que será de aplicación directa en toda la Unión Europea a partir del 25 de mayo de 2018.

La regulación establece que los datos de salud tienen un nivel de protección elevado basado en su categorización como Datos de Nivel Alto, que de manera general, está sujeto al consentimiento expreso del ciudadano y se le da la categoría de "datos de carácter personal relacionados con la salud". Esta categoría asegura:

- Protección de datos desde el diseño. *Se diseña un esquema seguro antes de implementar la solución.*
- Protección de datos por defecto. *Los datos almacenados tienen activos todo un conjunto de normas que los hacen seguros.*
- Medidas de seguridad. *Se establece la estandarización de seguridad que se debe cumplir*
- Mantenimiento de un registro de tratamientos médicos.
- Realización de evaluaciones de impacto sobre la protección de datos, *se debe presentar un esquema de periódico de pruebas de seguridad e impacto*
- Nombramiento de un delegado de protección de datos. *Cada empresa o institución deberá nombrar 1 responsable el cual vele porque se cumpla la norma internamente y será el responsable ante las auditorias del estado.*
- Notificación de violaciones de la seguridad de los datos. *Se articula un sistema de información para notificar y publicar las incidencias*
- Promoción de códigos de conducta y esquemas de certificación, *establece un programa de certificaciones obligatorias*
- Endurece el régimen sanciones con sumas de hasta 20.000.000€ o un 4% de la facturación global anual.

A modo resumen podemos ver la tabla 3 la cual plasma los beneficios mas importantes para sus principales destinatarios

Tablas 3. Beneficios de los cloud médicos

	Institución	**Pacientes**	**Profesionales de la Salud**
Gestión completa de la prescripción médica e historias clínicas	X	X	X
Control y planificación de agendas y turnos	X		X
Optimización de costos en prácticas y prescripciones médicas	X	X	X
Procesos automatizados que evitan errores humanos	X		X
Habilita la proactividad en el control de salud	X	X	
Incrementa la confianza en la institución		X	X
Información accesible oportunamente para la toma de decisiones	X	X	X

En Argentina no existentes aun leyes reglamentadas para el tratamiento y protección de los datos almacenados en *cloud* médicos. El marco legal de apoyo es el Registro Nacional de Bases de Datos, que registra empresas privadas y entes del estado que almacenan información general del índole sensible[13] de los individuos. Existe también en Argentina, la regulación de la transferencia internacional de datos personales y la prestación de servicios por parte de terceros en Cloud Computing (Ley 25.326 y el

decreto reglamentario 1558/2001), pero como en el caso anterior no hacen foco a los datos médicos del individuo, sino que es a nivel global de datos y su alojamiento en el exterior. Este vacío legal provoca poca confianza por parte de los consumidores del servicio y evitando el crecimiento del sector en el país.

3.3 Desafíos a enfrentar

Por lo que hemos analizado podemos destacar varios desafíos que se deben superar para el crecimiento de los cloud médicos

- Corregir las percepciones erróneas de los consumidores mediante campañas exhibiendo los beneficios de concentrar todos sus datos médicos en un solo punto y teniendo disponle la misma para cualquier profesional.

- Abordar las preocupaciones legítimas sobre la privacidad / seguridad.

- Proveer mayor información para que las organizaciones puedan evaluar el alcance, la complejidad, costos y tiempos de la migración versus las mejoras operativas que obtendrían y la calidad de información que administrarían.

- Mejorar el uso de recursos en las instituciones y evitar los abusos actuales a las prestaciones lo cual no está ligado a una baja en la calidad de la atención, sino que todo lo contrario una atención justa y de calidad.

- Crear ¿?? basados en EHR un *cloud* federal integrando SIISA y analizar las posibilidades reales de aplicación en Argentina, sumando el análisis de la legislación vigente para nuestro país.

-Generar premios e incentivos para profesionales e instituciones que se integren a investigar y aportar a este nuevo modelo en pos de mejora y crecimiento.

4. Conclusiones

Los *cloud* médicos tienen una potencialidad exponencial; según las estimaciones actuales entre el 2014 y el 2025 la demanda de este sistema crecerá entre 11 % a un 17 %, pero hacia el 2020 [9] la infraestructura actual no será suficiente para cubrir dicha demanda-

Para que el crecimiento siga de manera en exponencial la política pública debe acompañar con las leyes y normas necesarias para los países los cuales no las tienen y con esto dar el marco legal que necesita el consumidor para confiar en estos sistemas.

Sin lugar a dudas, son las organizaciones de salud los actores principales que pueden dar el impulso necesario a la implementación de *clouds* médicos, pero también deben enfrentar la falta de acompañamiento legal, los costos de inversión y la posible desconfianza que aún hay en los usuarios (en este caso pacientes y profesionales de la salud), acerca de este tipo de soluciones de TI.

Si bien estos factores, dificultan y lentifica el crecimiento de los *clouds* médicos, podemos identificar los siguientes aspectos positivos:

- Reducción de los costos de TI operativos al mismo tiempo que aumenta la escalabilidad y la flexibilidad de las soluciones de TI implementadas.

-Reducción de costos de atención a las instituciones

.-Reducción de cargos a las obras sociales en estudios y prácticas repetidas.

- Reducción y optimización de tiempos de atención permite a las instituciones implementar sistemas de información más rápidamente, reduciendo el tiempo necesario para la adquisición e implementación de soluciones, y reduciendo los desafíos de aprobación y financiación.

Disposición de tecnología móvil útil para usuarios alejados de los centros de salud o con movilidad reducida o impedida.

- Virtualización de lasas infraestructuras de EHR existentes para su disposición en el cloud.

Como vimos los cloud médicos no solo aportan ayuda médica, centralización y mejora de gestión asistencial, sino que generan y aportan al desarrollo sustentable , bajando costos al estado, a las obras sociales y optimizando los recursos de un país.

Conflicto de Intereses

“Los autores han declarado que no existen intereses en competencia”.

Referencias

1- Fundación openEHR,(2018), Open EHR. Londres, Reino Unido,http://www.openehr.org/es/home
2- Ministerio de Salud de la Nación (2017) Programa PICIS, Ministerio de Salud de la Nación,Buenos Aires, Argentina,http://www.msal.gob.ar/prensa/index. php/noticias/noticias-de-la-semana/418-la-informatizacion-del-sistema-de-salud-es-un-proceso-de-enorme-potencialidad
3- Onuae (2015) ,Estados Unidos, Wearables cuantificadores de salud, ventajas y futuro de

una arma de doble filo, http://ounae.com/wearables-cuantificadores-de-salud-ventajas-y-futuro-de-un-arma-de-doble-filo/

4- Cloud Standards Customer CouncilI (02/2017) Impact of Cloud Computing onHealthcareVersion 2.0 , Canadá,http://www.cloud-council.org/deliverables/CSCC-Impact-of-Cloud-Computing-on-Healthcare.pdf

5- Canada Health Infoway Inc.(2012) Revista Emerging Technolgy series, Cloud Computing in Health,https://www.infoway-inforoute.ca/en/component/edocman/resources/technical-documents/emerging-technology/545-cloud-computing-in-health-white-paper-full

6- Frost& Sullivan, (2016) California, Estados Unidos, Social Innovation in Healthcare, http://www.hitachi.eu/sites/default/files/fields/document/sib/whitepapers/social_innovation_in_healthcare_whitepaper1.pdf

7- Bhatt, Chintan M.,Peddoju, S. K.(08,2016) “Cloud Computing Systems and Applications in Healthcare”, https://books.google.com.ar/books?id=f4XvDAAAQBAJ&pg=PA50&lpg=PA50&dq=paper+cloud+health&source=bl&ots=ZCWS3dZftb&sig=g-MusORIOMkF8FjC0ehL_elV0-M&hl=es&sa=X&ved=0ahUKEwjktPfW3q3aAhUIHpAKHXqgBGQ4ChDoAQhIMAM#v=onepage&q=paper%20cloud%20health&f=false

8- Sistema Argentino de Información Jurídica (16,05,2018) http://www.saij.gob.ar/juan-cruz-gonzalez-allonca-cloud-computing-regulacion-transferencia-internacional-datos-personales-prestacion-servicios-parte-terceros-dacf150527-2015-10-01/123456789-0abc-defg7250-51fcanirtcod

9- Ministerio de salud de la Nación Argentina http://www.msal.gob.ar/observatorio/index.php/fuerza-de-trabajo/regulacion-del-ejercicio-profesional

10- International Data Corporation https://www.idc.com

Automatic and early detection of the deterioration of patients in Intensive and Intermediate Care Units: technological challenges and solutions

Javier Balladini[1], Pablo Bruno[1], Rafael Zurita[1], and Cristina Orlandi[2]

[1]*Universidad Nacional del Comahue, Neuquén, Argentina*
{javier.balladini, rafa}@fi.uncoma.edu.ar, pablo.bruno@est.fi.uncoma.edu.ar
[2]*Hospital Francisco Lopez Lima, Río Negro, Argentina*
orlandi.mariacristina@gmail.com

Abstract

In the Intensive and Intermediate Care Units of healthcare centres, many sensors are connected to patients to measure high frequency physiological data. In order to analyse the state of a patient, the medical staff requires both appropriately presented and easily accessed information. As most medical devices do not support the extraction of digital data in known formats, medical staff need to fill out forms manually. The traditional methodology is prone to human errors due to the large volume of information, with variable origins and complexity. The automatic and real-time detection of changes in parameters, based on known medical rules, will make possible to avoid these errors and, in addition, to detect deterioration early. In this article, we propose and discuss a high-level system architecture, an embedded system that extracts the electrocardiogram signal from an analog output of a medical monitor, and a real-time Big Data infrastructure that integrate Free Software products. We believe that the experimental results, obtained with a simple prototype of the system, demonstrate the viability of the techniques and technologies used, leaving solid foundations for the construction of a reliable system for medical use, able to scale and support an increasing number of patients and captured data.

Keywords: Intensive Care Unit, Clinical Decision Support System, Medical Rules Processing, Big Data, Embedded System.

Resumen

En las unidades de cuidados intensivos e intermedios de centros de salud, muchos sensores están conectados a los pacientes para medir datos fisiológicos de alta frecuencia. Para analizar el estado de un paciente, el personal médico requiere información presentada de manera apropiada y de fácil acceso. Como la mayoría del equipamiento médico no admite la extracción de datos digitales en formatos conocidos, el personal médico completa formularios manualmente. Esta metodología es propensa a errores humanos debido al gran volumen de información, con orígenes y complejidad variable. La detección automática y en tiempo real de cambios en los parámetros, basados en reglas médicas conocidas, permitirá evitar estos errores y, además, detectar el deterioro de forma temprana. En este artículo, proponemos una arquitectura de alto nivel del sistema, un sistema embebido que extrae la señal del electrocardiograma de una salida analógica de un monitor médico, y una infraestructura Big Data de tiempo real que integra productos Software Libre. Creemos que los resultados experimentales, obtenidos con un prototipo, demuestran la viabilidad de las técnicas y tecnologías utilizadas, dejando sólidas bases para la construcción de un sistema confiable para uso médico, y capaz de escalar para soportar un número creciente de pacientes y datos capturados.

Palabras claves: Unidad de Cuidados Intensivos, Sistema de soporte a la decición clínica, Procesamiento de reglas médicas, Big Data, Sistema embebido.

1 Introduction

In health centres, the Intensive Care Unit (ICU) provides comprehensive, rigorous, and continuous care for adult persons who are critically ill and who can benefit from treatment, providing a good die for unrecoverable patients. Some health institutions, in turn, have an Intermediate Care Unit (IMCU) that, unlike the ICU, provides care to patients who do not require life-sustaining therapeutic treatments, but to monitoring and control.

In both units, all patients are connected to a monitor that measures their vital signs and issues alerts that indicate a risk to their health. These alerts are determined under criteria based on the standard population. In the case of the ICU, patients could additionally be connected to other medical equipment. Some of them, such as mechanical respirators, also issue risk alerts. Almost all alerts issued by a medical equipment are

only based on parameters that it measures. Exceptionally, one device can be interconnected with another to issue alerts based on parameters measured by both devices.

In order to analyse the state of a patient, the medical staff require both appropriately presented and easily accessed information. As most medical devices do not support the extraction of digital data in known formats, medical staff need to fill out forms manually. These forms are used to record data observed in each equipment at hourly intervals, and to record data related to medical studies (as laboratory tests). At the end of each day, the physician analyse the data of the forms and produce indications for the nurses, such as modifications in the medication, practices to be performed on patients, etc.

Our objective is to develop a computer system that allows the automatic and early detection of deterioration of patients hospitalised in ICUs and IMCUs, through the real-time processing and analysis of digital health data. The data are acquired from different sources, including the real-time data extraction from medical equipment.

The traditional methodology is prone to human errors due to the large volume of information, with variable origins and complexity, that the medical staff must analyse. The automatic and real-time detection of changes in parameters (from multiple digital sources: software systems and medical equipment), based on known medical rules, will make it possible to avoid these errors and, in addition, to detect deterioration early. The latter will give physicians the ability to plan and begin treatments without delays, possibly increasing their effectiveness (and consequently reducing mortality) and decreasing their economic cost. Additionally, the accuracy of the diagnoses could be increased by contemplating the totality of the data generated by the medical equipment connected to the system, instead of a few manually registered. In the long term, the historical record of data will be extremely valuable for conducting studies to discover patterns in the data that can predict pathologies, such as septic shock [1].

In the literature, few systems of this type are found, some of them are [2, 3, 4, 5, 6, 7]. However, its authors have not exposed, to the community, accurate information on how they have addressed the different challenges inherent to the system and/or the performance achieved by their solutions, or they use proprietary software or hardware (for data acquisition), or they do not meet the requirements of: scalability, fault tolerance, and interoperability.

In this paper we discuss challenges related to the construction of the system and propose solutions that were implemented in our prototype. It is designed to be deployed at the Francisco Lopez Lima Hospital, a public hospital with relatively high financial constraints, located in the city of General Roca, Río Negro, Argentina.

The main contributions of this work are focused on the proposal of:

1. A high-level system architecture, which support multiple hospitals with integrated data storage and knowledge extraction, considering hospitals without Internet leased lines. It uses a mix of local computing and a public cloud (to reduce economic cost). An initial approach was proposed in [8].

2. An embedded device that extracts the electrocardiogram (ECG) signal from an analog output of a medical monitor, performs an analog-digital conversion, and transmits it via WiFi to the platform that process the signal. This device seeks to be an alternative to the problem of digital data extraction of medical equipment produced by the use of different communication interfaces, mostly proprietary, and whose specifications are not published by the manufacturers.

3. A real-time Big Data infrastructure that, based on streams of signals data and other health data, allows to process rules to determine and issue alerts indicating risk in the health of patients. A distributed, scalable, fault tolerant and interoperable solution is proposed, based on the integration of software products under Free Software licenses. It is included a web system that allows user interfaces with visualisation of signals and alerts in real-time.

The rest of this paper is organised as follows. Section 2 describes the high-level architecture of the system. Section 3 presents the embedded system that acquire the ECG signal from a medical monitor. Section 4 discusses the Big Data infrastructure that support real-time processing, analysis, and visualisation of health data. Finally, section 6 presents the conclusions and future works.

2 High-level System Architecture

Figure 1 shows the high-level system architecture proposed. In this figure it is observed several Hospitals connected to a Public Cloud. As we are considering hospitals with unreliable Internet access, the critical part of the system requires to perform on a dedicated computing system placed inside each hospital. In a hospital, the component Data Acquisition is in charge of the connection between our system and external health data sources: Medical Equipment and the Electronic Health Record (EHR) System. The Data Acquisition module streams the data to a processing infrastructure. The infrastructure must be capable of processing, in real time, a large volume of data per unit of time. For that reason it receives the name of

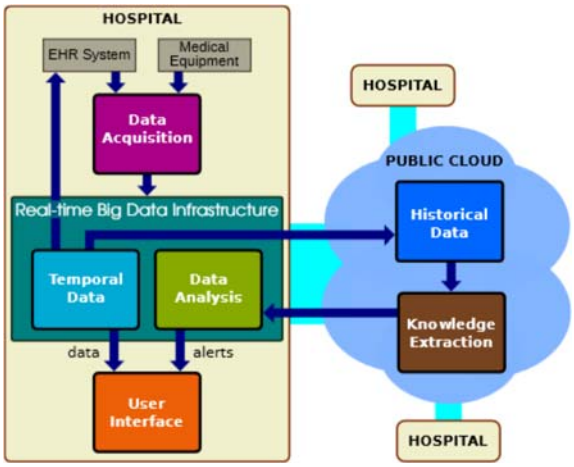


Figure 1: High-level System Architecture

"Real-time Big Data Infrastructure". This large volume of data per unit of time comes mainly from signals produced by medical equipment. For example, one of these signals is the electrocardiogram (ECG), a test that records the electrical activity of the heart over a period of time using electrodes placed on the pacient skin, and is used to detect cardiac problems. A typical vital signs monitor takes 1,000 samples per second, and advanced converters can sample at 10,000 to 15,000 per second or even higher [9].

The Real-time Big Data Infrastructure must manage Temporal Data (see Figure 1), which are health data relevant to the system, corresponding to patients currently hospitalised. All data acquired from the time of the patient's admission to the ICU/IMCU are considered. When the patient leaves the inpatient unit, some data may be recorded in the EHR system.

Some data are useful by themselves, and other data may need to be analysed to produce new data. For example, the signals obtained from medical equipment could be interpreted through a process of Data Analysis (see figure 1), generating meta-data, that is, new health data of patients. This is the case of parameters such as heart rate, which comes from the analysis of the ECG. The system also performs another type of data analysis: the processing of medical rules. In the figure 2 two example rules are shown. Each rule defines the conditions, relating parameters and values, which must be met to generate an alert indicating risk in the patient's health. Each patient can be associated with a particular set of rules. Alerts are classified according to a risk coefficient that determines their importance; the lower the value, the greater the risk to patient health and, therefore, the greater the importance of the alert. Each alert has a description, possibly different, for physicians and nurses. One of both alert descriptions could be null, allowing to support alerts directed to a single group, physicians or nurses. Optionally, each of these groups is associated with a treatment plan to be carried out. The rule also determines the activation or deactivation of other rules (including itself), possibly after a certain time. Once an episode of deteriorating patient health is overcome, certain rules are re-activated.

The User Interface allows nurses and physician to receive alerts, and to visualise any Temporal Data such as raw data (signals, vital signs, etc.) or statistical processed data (tables, charts, etc.). It is possible to view current raw data (in real-time) and to explore previous data.

Our system includes the storage of Historical Data for Knowledge Extraction, useful for future medical research and the discovery of patterns for predicting pathologies. The Knowledge Extraction is directed by physicians, and the output of this component are clinical rules and data specifically tailored to enable physicians to perform clinical research. The new rules are then incorporated to the set of rules used for Data Analysis at each hospital.

The Historical Data includes all acquired inpatient data from any hospital. This data is saved in a persistent way in a Public Cloud. It must be observed that the temporal storage on hospitals needs to save only data of current inpatients. This means the storage does not need to scale simply by the passing time, but only when grows the amount of beds or the number of considered health parameters (possibly with new acquired data from medical equipment). Instead, the persisting storage requires to scaling with the passing time because it must save the new medical data that are received from hospitals. Therefore, the scalable storage offered by a public cloud is very useful. The computing resources of the public cloud can be used to perform the Knowledge Extraction. As intensive computation for Knowledge Extraction need to be done sporadically, a Public Cloud will be effective in cost (avoiding the high costs of a dedicated system and technicians).

3 Data acquisition

The acquisition of data from an EHR system does not present significant technological problems because they are normally prepared for interoperability. The challenge is in the data acquisition from medical equipment, the lowest level component (hardware and software) of the whole system presented on this work. It must collect signals from medical monitors (INPUT), and send the digital information (OUTPUT) to the Real-time Big Data Infrastructure, to be processed. If analog signals exist a conversion to a proper digital representation is required before the transfer.

Interfacing with the complete set of equipments is the long term goal for this level. Unfortunately, the hospital keep using a wide and complex variete of ancient and newer monitoring equipments, so different INPUT/OUTPUT interfaces exist. In some cases, two models from a same manufacturer (but different year of

```
RULE 1
IF (RR > 30 per minute)
   and (Hyperlactacidemia:  > 3 mmol/L)
   and (Arterial Hypotension:  SBP < 90 mmHg or MAP < 70 mmHg)
   and (Fever > 38 ℃)
THEN
   Risk coefficient:  2
   Nurse alert:  Probable Septic Shock
   Nurse treatment:  Infuse fluid 20 ml/Kg
   Physician alert:  Probable Septic Shock
   Physician treatment:  Two blood cultures and start antibiotic therapy
   Activate rules:  2 after 30 minutes
   Deactivate rules:  1

RULE 2
IF (Arterial Hypotension:  SBP < 90 mmHg or MAP < 70 mmHg)
THEN
   Risk coefficient:  1
   Nurse alert:  Septic Shock
   Nurse treatment:  Start noradrenaline infusion:  16mg/250cc dextrose 5% 21ml/Hour ⇒ Objective MAP 70 mmHg
   Physician alert:  Septic Shock
   Physician treatment:  no
   Activate Rules:  3
   Deactivate Rules:  2

References:
   RR = Respiratory Rate
   SBP = Systolic Blood Pressure
   MAP = Mean Arterial Pressure
```

Figure 2: Examples of medical rules

production) do not use identical hardware ports and/or software protocols for the output signals. Thus the data extraction from medical equipment raises a research topic previous to the design of an embedded system for data acquisition, because internal specifications for those variety of interfaces are not always published by the manufacturers. Either since it uses propietary protocols, or the proper documentation might not available (if, e.g. an ancient equipment is not supported anymore). As a consequence, the data acquisition might not be completely possible for all the equipments.

Many monitors uses the RS-232 standard, for the electrical and mechanical characteristics of outputs. Differences here ocurres on the internal data level. RS-232 is commonly used for serial communication betweens systems, but some equipment might work internally with other different protocols, for interconnecting devices from the same vendor only.

In order to develop and test our first data acquisition embedded system prototype we choose, for interfacing, the Life Scope LC, BSM-3101, from Nihon Kohden. The BSM-3101 is a medical monitor, with an analog interface for continuos ECG data output. Since it features a non-digital output (which requires to be converted) this equipment is suitable for testing the longest use case (list of actions) of our data acquisition prototype. Interfacing other digital outputs monitors might be straighforward (if the correct documentation is available).

3.1 Embedded ECG Signal Acquisition System

The data acquisition system prototype architecture is shown on figure 3. It comprises a microcontroller and a single board computer (SBC). The former is a 8-bit

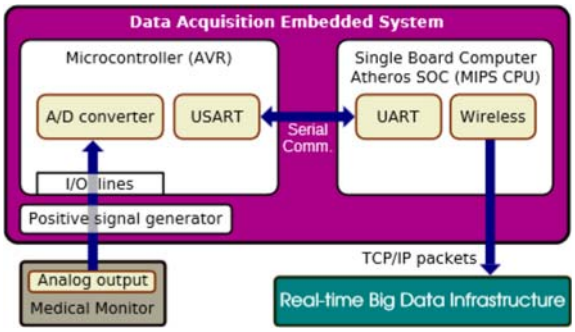


Figure 3: Data Acquisition Embedded System Architecture

CPU (AVR architecture) with several low level I/O lines, set for interfacing with monitors. It also features 2KB SRAM, 32KB flash memory, a 6-channel 10-bit analog-to-digital converter, SPI serial port, a two-wire serial interface, and a serial programmable USART. The selected SBC includes a Wi-SOC from Atheros (32-bit 400Mhz MIPS CPU, 32MB RAM, 4MB flash memory) with low power consumption and reliable Wi-Fi interface. The Wi-SOC is the wireless communication bridge between the whole data acquisition device and a central server.

The two components communicate using the Universal Asynchronous Receiver/Transmitter (UART). The maximum bits/baud rate per second is 115,200, which represents almost 100kbits per second. In case of there is an excessive continous data input, the UART would be the limiting hardware on this architecture. However, it is planified to use just one of this low cost data acquisition device per patient/bed, so there should not be greater input data than the limit imposed by the UART.

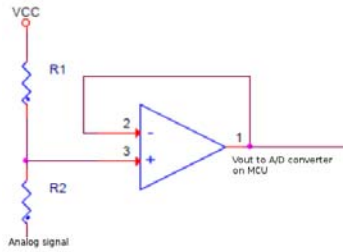


Figure 4: Positive signal generator

On the software level, a custom firmware for the microcontroller was developed. It reads the analog signal from ECG using an analog-to-digital converter driver. Then, after the conversion, the MCU transfers the 10-bit resolution digital value to the SBC using an UART driver. Since there is not other software tasks so far, the INPUT action is accomplished using polling programming, inside an infinite loop. When some input data is available, the infinite loop calls the send procedure, which is part of the UART driver.

It was observed (when studying the ECG monitor internals) that the analog output has positive and negative voltages (-5v to 5v range). In view of the fact that the choosen MCU is not able to read the negative voltage (the ADC works on 0v to 5v range) the analog signal was mounted on a unipolar positive signal generator little circuit, which is achieved using a voltage divisor and a operational amplifier (OpAmp). Figure 4 shows this circuit.

The software in the SBC includes two main components: a custom Linux kernel (featuring UART and wireless drivers, and TCP/IP software layers), and the userspace software. The latter was built using buildroot project, which is suitable for small Linux devices with low memory. On userspace there is also a custom application, which uses the Linux UART driver to read for incoming digital data from MCU. When digital data bytes are read the userspace application transfers those to the central server, using a TCP/IP connection.

3.2 ECG Signal Acquisition System Validation

Several measurement tests of rate and precision were performed in a real environment at the hospital. It is known that thousand samples per second from the ECG are an adequate amount for describing the patient condition on real-time. This rate of data was taken for several hours on this actual environment with no lost of information, and a wireless TCP/IP online communication during the whole test.

The most important validation is about the precision of data acquired. For this purpose a comparison between results obtained by the prototype and a real oscilloscope (PicoScope 2203) was made. Many samples for several seconds were taken using both data acquisition systems, at the same moment. All the samples were saved, and a script was used to graphically show the representative curves (useful for medical diagnosis, and for our validation). When both curves were overlapped it was demostrated that the prototype is acquiring the data from ECG correctly. Figure 5 shows the ECG over 2 sec, with the two set of samples graphically overlapping. The green curve (on top of the violet curve) are the samples taken by our prototype. The violet curve the samples gotten by the PicoScope.

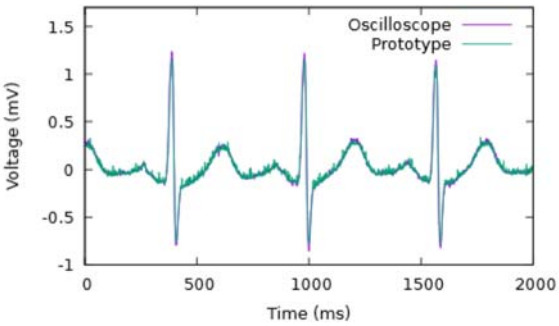


Figure 5: ECG prototype validation

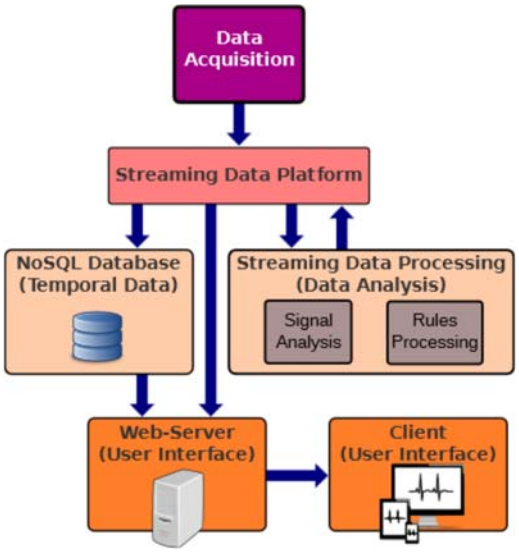


Figure 6: Architecture of the Real-time Big Data Infrastructure

4 Real-time Big Data Infrastructure

The Real-time Big Data Infrastructure, whose functionality was described in section 1, is implemented using the architecture shown in figure 6. Data are organised in a central platform, the Streaming Data Platform, which receives data streams and makes them available to other components to be consumed in real time. It works as a messaging system or message queue, under the publication-subscription pattern. This organisation of the data allows to simplify the flow of communications between the different components, producing a

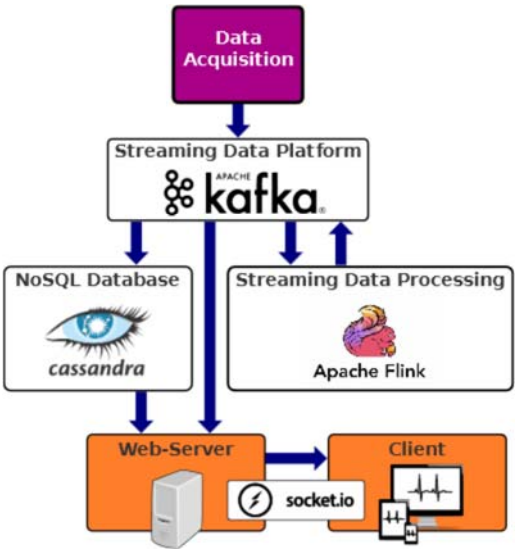


Figure 7: Technologies of the Real-time Big Data Infrastructure

low coupling between them.

The Data Acquisition module extracts data from the EHR System and the Medical Equipment, and sends them in the form of streams to the central platform. Most of data received by the platform comes from the Data Acquisition.

Streaming Data Processing consumes data from the central platform, and is responsible for performing a Data Analysis, that is, the analysis of physiological signals and the processing of medical rules. The results of the processing/analysis are returned to the central platform.

The NoSQL Database allows the storage of Temporary Data. This type of database (NoSQL) are designed to store and process big data, with high-performance reading and writing operations [10]. The NoSQL Database consumes data from the central platform (raw data produced by the Data Acquisition and data generated by Streaming Data Processing) and writes them to secondary storage.

Finally, the User Interface module presents the data (signals, vital signs, etc.) to the physicians and nurses, possibly making a small prior processing of them when statistical data are required. This module can receive data from the Streaming Data Platform or NoSQL Database, depending on whether the required data are real time or past time, respectively.

The figure 7 shows the software products, with Free Software licenses, selected for the implementation of the Real-time Big Data Infrastructure. The following sections describe the operation of each used software, and how they are integrated into our prototype.

4.1 Streaming Data Patform

The Streaming Data Platform is implemented by Apache Kafka, a distributed streaming platform that

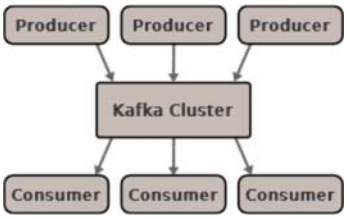


Figure 8: Kafka Producers and Consumers

handles data streams in real time [11]. Kafka was originally developed at LinkedIn and now is part of the Apache Software Foundation.

The platform allows scaling to multiple nodes of a cluster, allowing us to easily support the increase in the number of patients and in the volume of data per patient (especially when new signals will be acquired from medical equipment). In addition, it is tolerant to failures, an essential property for a critical application of the health field.

The interaction with Kafka is carried out through subscriptions/publications of "streams of records" (representing data streams). Thus, there are producers who make publications, to send streams of records to the platform, and consumers who make subscriptions to receive streams of records from the platform. This is exemplified in figure 8. Specifically for our prototype, the Data Acquisition module is a producer, Apache Flink is both consumer and producer, and Cassandra and the web server are consumers.

Each record (of a stream) consists of a key and a value. The streams of records are stored in categories called topics. For each topic there is a log, which stores the records of the topic. A topic can subscribe zero, one or more consumers, who will read the desired records (the most recent or past) from a single shared log. The logs are maintained persistently, and are deleted after a specified time of life (whether their records have been consumed or not). The use of writing in the filesystem does not involve a loss of performance because Kafka has pagecache-centric design.

A log can be partitioned. Each partition can be stored in a different node of the cluster, and a partition will only be in one node. Partitioning allows to use the storage of more than one node for the same log. In addition, it allows to increase the performance of the system by means of parallel access to the log (from multiple nodes). Kafka only provides a total order over records within a partition, not between different partitions in a topic. A global order of topic records can be achieved using a single partition topic. However, if the use of multiple partitions are required, a solution can be found by determining which records are assigned to each partition (based on the key) at the producer.

Kafka replicates its partitions over multiple nodes for fault-tolerance. Each partition has one node acting as leader and zero or more nodes acting as followers. The leader handles all read and write requests for the

Topic	Key	Value
Alerts	<patient_id>	<time_sec>, <risk_coefficient>, <nurse_alert>, <nurse_treatement>, <physician_alert>, <physician_treatement>
ECG_1	<time_msec>	<sample>
VitalSigns_1	HR	<sample>
VitalSigns_1	RR	<sample>
⋮	⋮	⋮
ECG_*n*	<time_msec>	<sample>
VitalSigns_*n*	HR	<sample>
VitalSigns_*n*	RR	<sample>

Table 1: Kafka topics of our prototype

partition. At the same time, the followers passively replicate the leader. If the leader fails, one of the followers will automatically become the leader. For load balance purpose, each server acts as a leader for some of its partitions and a follower for others.

In our prototype, on the one hand, a unique alert topic has been created for all patients. Thus, each consumer of alerts (the web server and Cassandra) will make a single subscription to Kafka to receive all alerts, whatever the patient. On the other hand, each patient is identified with a number ranging from 1 to the total of possible inpatients. The following topics are defined per patient: a topic for the ECG signal and a topic for the Vital Signs. In table 1 the content of each topic implemented in our prototype is shown, where *n* is the maximum patient *id*, HR is the heart rate and RR is the respiratory rate. All topics have been defined with a single partition. This allows to preserve, in a simple way and without any detriment, the global order of the records of the streams.

The criterion for determining topics for data coming from the Data Acquisition module, is as follows. A topic groups different parameters when two conditions occur: the measurements of the parameters are made at low frequency, and normally the parameters are required together (by Apache Flink, Cassandra, or the web server).

4.2 Stream Data Processing

Stream Data Processing performs two activities: signal analysis and medical rule processing. This module is implemented by Apache Flink [12], a stream processing framework to create distributed, scalable, low latency, and fault tolerant applications. Other frameworks offer similar solutions but using microbatching techniques (like the well-known Apache Spark with Spark Streaming). Unlike them, Flink was created with Streaming Processing in mind, allowing the processing of individual elements of a stream with very low latencies.

Flink works only with data in main memory. Therefore, it is necessary that all data fit in this memory. Fortunately, Flink implements its own memory management inside the Java Virtual Machine (JVM), with less garbage collection overhead. Furthermore, Flink can scale to several nodes of a Cluster (or Cloud), allowing the use of more main memory, and the performance improvement through parallel computing.

Flink supports fault tolerance through checkpoint-restart mechanism to consistently recover the state of the distributed streaming dataflow under failures. The checkpoint can be stored in a configurable place, possibly using a distributed file system. In case of a program failure (due to a failure in software, in computer hardware, or in the network), Flink stops the distributed streaming dataflow. The system then restarts from the last successful checkpoint. As our application has a small state, the checkpoint is very light-weight and can be done frequently with low impact on performance. It is necessary that the checkpoint interval of Flink be consistent with the retention time configured for the Kafka logs. In another case, the recovery will not be complete.

Currently, in our prototype, signal analysis is done with an application written in C language and its integration into Apache Flink is under development. So, at this time, we use Appache Flink only for the processing of medical rules.

Basically, Flink programs are composed of the following 3 parts. *Data source* is the incoming data to be processed. *Transformations* is the processing step, that is, the modifications on the incoming data. Finally, *Data sink* is where Flink sends data after processing.

Particularly to our system, each part is performed as following:

Data source: Flink makes a subscription to Kafka for each patient *id* (from 1 to the maximum number of inpatients) in the topics ECG_*id* and VitalSigns_*id*.

Transformations: As data is received from Kafka, Flink analyses if conditions specified in rules (associated with each patient *id*) are met.

Data sink: When conditions of a rule are met, an alert is issued by producing a new record in the Alerts topic of Kafka.

Flink is natively prepared for integration with Kafka, so it is not necessary to add a special connector between both platforms.

Flink supports the kind of processing required for medical rules. Flink's Complex Events Processing library (CEP) is of special interest for our purpose. With it, Flink is able to process information by detecting patterns that occur in the data, also called events, and then to produce some output. The CEP library has a Pattern API. This API provides tools to detect sequences of

patterns that can be extracted from the input stream. This sequence can also be seen as a graph where each node is a simple pattern and the transitions are made through the fulfillment of a specified condition.

4.3 NoSQL Database

During the time that a patient is hospitalised, it is vital to store all your health data (including alerts) to be consulted by nurses and doctors. These data need to be written in secondary storage and without any compression for quick access. Once a patient leaves the inpatient unit (ICU/IMCU), their data are removed from the local system. However, before being eliminated, data need to be stored (possibly compressed) in the Public Cloud. This Historical Data will be used for Knowledge Extraction. In addition, some data may be recorded in the EHR system.

A database is needed to store medical data generated for each patient during his hospitalisation. The high-frequency of signal data (such as the ECG), multiplied by the number of patients, will produce a very high number of insertions in the database. In turn, as they occur, it is required to respond quickly to queries originated by the web server. The NoSQL databases are appropriate for these requirements and, within existing ones, we choose Cassandra [13].

Regarding the data model, Cassandra's philosophy is to create optimised tables for certain queries and to not implement expensive operations such as joins. Instead, it opts for data redundancy. Cassandra works in a distributed manner and is fault-tolerant. Replication in different nodes allows low latency operations.

Cassandra does not have native connection with Kafka. However, Kafka provides Kafka Connect, a means for integration with other systems through the creation of connectors. There are two types of connectors: the Source Connectors, which import data of a system and insert them into one or more topics (acting in a similar way as a producer) and the Sink Connectors, which export Kafka information to a target system. The latter allows Cassandra to be connected with Kafka, and to be updated as data from the Kafta topics (Alerts, and ECG and VitalSigns for each patient) are ingested.

4.4 Web Server

The User Interface module presents patient data to doctors and nurses. The data involves: alerts, signals, vital signs, and any other medical data. The data can be presented in real time, and in that case the data need to be extracted from Kafka. In addition, it may be necessary to present past-time data. For example, a doctor or nurse might check a patient's ECG and temperature curves, which occurred minutes ago, or at night. In this case, data need to be extracted from Cassandra.

	System Node	Support Node
Processor	1 x Intel Xeon E5-2630	1 x Intel Core 2 Quad Q6600
	6 cores, 12 threads	4 cores, 4 threads
Main Memory	16 GB	8 GB
kernel version	Linux Debian 4.9.18-1	Linux Debian 4.9.82-1

Table 2: Characteristics of the experimental platform

The User Interface is implemented by a web server and web or app clients. The server connects to Kafka (using a Kafka API) and subscribes to the topics of interest, to receive data in real time. When it is required to access past time data, the web server queries Cassandra.

The data from the client interfaces should be updated as the server receives data from Kafka. The typical polling technique (in which each required data need to be requested) is not appropriate for this situation. On the contrary, once a client has requested a certain data stream to the web server, data should flow continuously. To carry out this type of client-server communication, the WebSockets protocol (defined in RFC 6455) can be used. This provides full-duplex communication channels over a single TCP connection. Through a channel, the client can make requests or send data to the server. In turn, by another channel, the server can send data to clients, without request for them constantly. In our prototype we use the Socket.IO library [14], an implementation of WebSockets with extra features.

4.5 Experimentation

The objective of the experimentation is to determine if a server of modest characteristics could support the processing for patients at the ICU (with 7 beds) and IMCU (with 5 beds) of the Francisco Lopez Lima Hospital. The prototype implements the Real-time Big Data Infrastructure using a single node.

A support node is used for:

- Emulate the Data Acquisition module: data streams are generated by Python scripts.
- Capture the alerts: a consumer of Kafka, implemented in Python, receives the alerts.
- Take measures for performance evaluation.

The performance evaluation of our prototype consists of determining the minimum, maximum and average latency to issue alerts. The latency time of an alert is measured from the moment the last necessary data that causes the alert is sent by the additional node, until the alert reaches the additional node.

The characteristics of each node used for experimentation are shown in table 2. Both nodes are connected to a local network of 1 Gbps.

Experiments were performed for 8 and 20 patients. For each patient it is generated an ECG signal with a frequency of 1 sample per millisecond, and 5 vital

	8 patients	20 patients
Minimum (msec)	18	29
Average (msec)	154	167
Maximum (msec)	290	323

Table 3: Alert issue latency

signs with a frequency of 1 sample per second. Each experiment runs for 5 minutes, and every 15 seconds an alert per patient occur. Table 3 shows the minimum, average and maximum latency required to issue alerts. The result obtained allows to determine that the prototype, running on a modest server, is suitable for use in the FLLH. However, it is necessary to use more than one node for the system to be faul tolerant.

5 Conclusions and Future Works

Our objective is to develop a computer system that allows the automatic and early detection of deterioration of patients hospitalised in ICUs and IMCUs, through the real-time processing and analysis of digital health data. In this article the challenges and the proposal of solutions that we implemented in a prototype were discussed. The prototype was developed and evaluated to be used in a public hospital of Argentina.

The general problem of ICUs/IMCUs was presented. We have described a high-level system architecture which supports multiple hospitals without Internet leased lines. The solution uses a computing system at each hospital and a Public Cloud, used to store historical data and for knowledge extraction. The difficulty of extracting data from medical equipment using unknown interfaces (hardware and software) was discussed. We have presented a solution based on an embedded system that we develop for acquire the electrocardiogram (ECG) signal from an analog output of a medical monitor, performs an analog-digital conversion, and transmits it via WiFi to the platform that process the signal. We have detailed a real-time Big Data infrastructure that, based on streams of signals and other health data, allows to process rules to determine and issues alerts indicating risk in the health of patients. The infrastructure is distributed, scalable, fault tolerant and interoperable, based on Free Software products.

We believe that experimental results demonstrate the feasibility of the techniques and technologies used, leaving solid foundations for the construction of a reliable system for medical use, able to scale and support an increasing number of patients and captured data.

As future works, different fault tolerance configurations will be evaluated. The detection of QRS complexes of ECG signals is expected to be integrated into the prototype. Furthermore, it is planned to incorporate the detection of anomalies in ECG signals to avoid the contamination of the system with erroneous data. It is necessary to acquire new signals: oxygen saturation in blood, body temperature and blood pressure. Interconnection with mechanical respirators is also of interest. Finally, the research will be directed to the knowledge extraction module, used to define rules for pathologies prediction.

Acknowledgements

We thank Ariel Stancato for his collaboration in the design of the unipolar positive signal generator circuit.

Competing interests

The authors have declared that no competing interests exist.

References

[1] A. Bravi, G. Green, A. Longtin, and A. J. Seely, "Monitoring and identification of sepsis development through a composite measure of heart rate variability," *PLoS One*, vol. 7, no. 9, p. e45666, 2012.

[2] S. Balaji, M. Patil, and C. McGregor, "A cloud based big data based online health analytics for rural nicus and picus in india: Opportunities and challenges," in *Computer-Based Medical Systems (CBMS), 2017 IEEE 30th International Symposium on*, pp. 385–390, IEEE, 2017.

[3] "ehcos smarticu." Available at: `https://www.ehcos.com/productos/ehcos-smarticu/`. Accessed on 2018-06-09.

[4] "Excel medical." Available at: `http://excel-medical.com`. Accessed on 2018-06-09.

[5] B. R. Matam and H. Duncan, "Technical challenges related to implementation of a formula one real time data acquisition and analysis system in a paediatric intensive care unit," *Journal of clinical monitoring and computing*, pp. 1–11, 2017.

[6] "Amara health analytics." Available at: `http://www.amarahealthanalytics.com`. Accessed on 2018-06-09.

[7] J. Antony, "A tablet pc based system for ubiquitous patient monitoring and smart alert generation in an intensive care unit," *International Journal of Computer Applications*, vol. 67, no. 6, 2013.

[8] J. Balladini, C. Rozas, F. E. Frati, N. Vicente, and C. Orlandi, "Big data analytics in intensive

care units: challenges and applicability in an argentinian hospital," *Journal of Computer Science & Technology*, vol. 15, 2015.

[9] P. Kligfield, L. S. Gettes, J. J. Bailey, R. Childers, B. J. Deal, E. W. Hancock, G. van Herpen, J. A. Kors, P. Macfarlane, D. M. Mirvis, O. Pahlm, P. Rautaharju, and G. S. Wagner, "Recommendations for the Standardization and Interpretation of the Electrocardiogram: Part I: The Electrocardiogram and Its Technology: A Scientific Statement From the American Heart Association Electrocardiography and Arrhythmias Committee, Council on Clinical Cardiology; the American College of Cardiology Foundation; and the Heart Rhythm Society Endorsed by the International Society for Computerized Electrocardiology," *Circulation*, vol. 115, no. 10, pp. 1306–1324, 2007.

[10] A. B. M. Moniruzzaman and S. A. Hossain, "Nosql database: New era of databases for big data analytics - classification, characteristics and comparison," *International Journal of Database Theory and Application*, vol. abs/1307.0191, 2013.

[11] "Apache kafka." Available at: `https://kafka.apache.org`. Accessed on 2018-06-09.

[12] "Apache flink: Scalable batch and stream data processing." Available at: `https://flink.apache.org`. Accessed on 2018-06-09.

[13] "Apache cassandra." Available at: `https://cassandra.apache.org`. Accessed on 2018-06-09.

[14] "Socket.io." Available at: `https://socket.io/`. Accessed on 2018-06-09.

Cloud computing application model for online recommendation through fuzzy logic system

Elham Shojaei[1], Emilio Luque[1], Dolores Rexachs[1], and Francisco Epelde[2]

[1]*Computer Architecture Operating Systems,Universitat Autnoma de Barcelona, Bellaterra Barcelona,08193, Spain*
Elham.Shojaeie@-campus.uab.cat
[2]*Medicine Department, Hospital Universitari Parc Tauli, E-08208 Sabadell, Barcelona, SPAIN*
fepelde@tauli.cat

Abstract

Cloud computing can offer us different distance services over the internet. We propose an online application model for health care systems that works by use of cloud computing. It can provide a higher quality of services remotely and along with that, it decreases the cost of chronic patient. This model is composed of two sub-model that each one uses a different service, one of these is software as a service (SaaS) which is user related and another one is Platform as a service (PaaS) that is engineer related. Doctors classify the chronic diseases into different stages according to their symptoms. As the clinical data has a non-numeric value, we use the fuzzy logic system in Paas model to design this online application model. Based on this classification, patienst can receive the proper recommendation through smart devices (SaaS model).

Keywords: Cloud computing, health care system, heart failure, fuzzy logic system.

1 Introduction

Demographic results show that both developed and developing countries are facing ageing population and as we expect with advancing age, chronic conditions will be more obvious, so it will detract from the quality of care. In general, the healthcare system tries to reduce costs and make better use of resources in minimum time while talking about improving health care system efficiency.
Improving quality is related to patient satisfaction, so its needed to reduce wait times and annoying delays for patients. Chronic patients often need to access healthcare systems and many of them need to be readmitted even though they are not in an emergency or dangerous situation. Likewise, many chronic diseases are preventable or predictable. Many chronic conditions are relevant to lifestyle choices that are under our control. Reducing unnecessary attendance of chronic patients to the health care system and controlling their visiting time could be an important solution to improveing healthcare system efficiency.
Using technology and smart devices to monitor and control chronic patients remotely can be a great help to reduce the burden of visitors and enable us to manipulate discipline in the health care system [1],[2].
So we will gain better patient management and consequently better quality of services. Thus, offering an e-health care system can reduce complications due to chronic conditions with efficient follow-up, and provide online environmental monitoring of patients.
To have secure, accessible, flexible and economical services, we use cloud computing to design the proposed model. The model enables transmission of a patients health status, vital signs and risk factor data remotely from the respective healthcare setting. So the patient is going to be monitored remotely and the systems will be updated by receiving the patients data. Health care staff can access the patient's data when they need to improve information, knowledge and this leads to a better quality of care [3] [4].
Heart Failure is one leading causes of death among the chronic diseases. The prevalence of heart failure depends on the definition applied, but it is approximately 1 to 2 % of the adult population in developed countries, rising to $\leq$ 10% among people over 70 years of age. Significant heart failure signs and symptoms can be simply measured by the patient remotely and be sent to clinical severe [5]. This two reason, made heart failure prior to other diseases for use to work on that. Usually, clinical data are many-valued data and it is not in the precise value of zero or one, in other words this data is close to human language. That is why we use the fuzzy logic system to design our proposed model.
This research is carried out with the collaboration of healthcare staff at the Emergency Department of Hospital Universitari Parc Taul (one of the biggest hospitals in Catalonia, Spain) which provides care service

to a catchment area of about 500,000 people, receiving over 160,000 patients annually in its ED[6].We have designed a model to consider how a heart failure patient can be monitored online through a cloud, and how a change in clinical parameters can modify the stage of health plus quality of care system, Then we are going simulate this model to evaluate and organize diverse visions and aspects of the structure model. So simulation provides us with this possibility to qualify the proposed model more safely [7].
This paper is organized as follows: the second secsion is about the steps in the design of the E-health model, in section 3 we give an overview about the E-health model through a cloud, in section 4 we talk about Fuzzy logic process, section 5 will deal conclusions and in section 6, we talk about competing for interest and at the end there are an acknowledgment.

2 The steps in the design of the E-health model

Digitizing healthcare information and communication provides stronger access to various health care services for patients and it speeds up service delivery and information exchanges. As we see it, E-health is a solution to improve healthcare system efficiency. So we need to implement an E-health system. In order to achieve a successful implementation, first, we need to design a model and increase the theoretical approach. This provides a better explanation and understanding of the strengths and weaknesses of the system. This model is designed based on the data in the real system. Data helps to analyze the problem of the current system, but in order to see how and why this suggested model succeeds or fails, we use simulation before implementation. This is because firstly, simulation gives us this chance to consider a different scenario with real or virtual data. Secondly, we can have a deep look at a different aspect of the designed model, thirdly, it can minimize our spending in time and finally, it is affordable and easy to use. The next step is analyzing the result that is gained by simulation. This result can help to evaluate the proposed model and predict the best model for implementation. All these steps are shown in Figure 1. In this study, our E-health model is composed of two sub-model, one of which is the patient model that determines the level of danger of the patient and the other is the online recommendation model that sends some recommendations to a patient based on their level of danger.

3 The components of the E-health model through cloud

In the online monitoring system, the amount of vital and clinical data is huge and it is expanding so fast. Furthermore, it needs to be available 24/7. A space storage or devices that can save all this data or

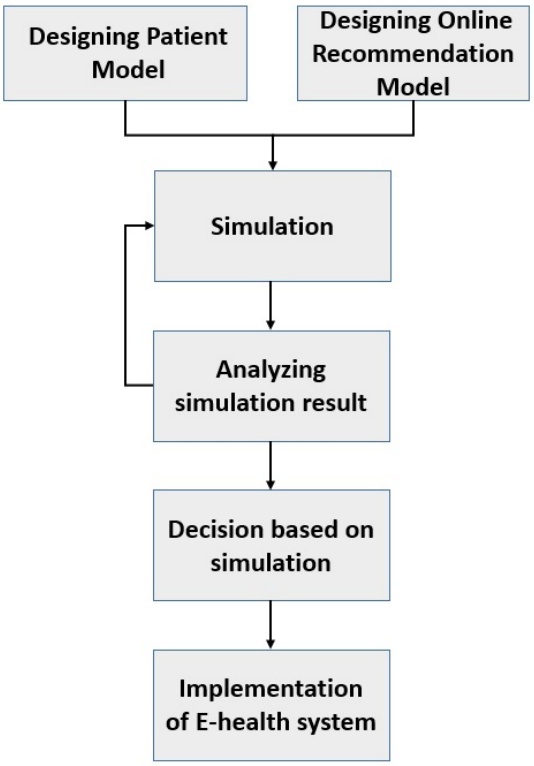


Figure 1: Steps design of E-health system

provide the services can be costly or slow. All these matters lead us to bring up the issue of cloud computing. One characteristic of cloud computing is a pay-per-use basis or using a subscription fee [8], the latter of which has the potential to be extended based on the demand and consumption. Cloud can provide us with a wide range of medical data, application and services anytime, anywhere. Cloud Computing provides a strong infrastructure and offers distance services for a health care system over the Internet. Figure 2 shows the general idea of the suggested cloud computing model, which is composed of two service models as following[9].

- Software as a Service (SaaS): This service is used as a Web interface that is designed as follows :
  Inputs: patient monitoring, historical data of the patient
  Process: prediction of patient evolution
  Outputs: NYHA Classification (figure 4), recommendations to the patient and to the healthcare staff
- Platform as a Service (PaaS): This is a development of the simulator that is achieved by a collaboration doctors and engineers

The health care staff and the patient can connect to cloud over the internet and use the proposed application. The patient can send clinical data and receive recommendation or an appointment, concurrently healthcare staff can store and access a patient's data. All of

this intercommunication happens in SaaS. PaaS determines how this intercommunication happens.

3.1 Software as a Service (SaaS):

SaaS provides Cloud-based software solutions (e.g. clinical systems) where consumers such as healthcare providers receive access to the software capabilities of the cloud. This layer is user-related, meaning it is accessible to users. In this model, the users are health care staff and patients. E-health care systems focus on clinical parameters related in order that the health care organizer can treat the chronic patient by the healthcare organizer. Chronic patients require follow-up to improve their adherence to treatment (compliance with the recommended preventive measures), and to evaluate changes in their pathology. The data is gained through monitoring or by being entered by the patient or healthcare staff and can be used for part of the treatment and sometimes for evaluating a diagnosis of the patients condition [10],[4].
As we can see in Figure 3, in the SaaS model of the e-health care system after patient succeed to connect to system over the internet, their data is considered and the problem is analyzed. If a patient is in a danger situation an urgent appointment is made for then. Otherwise according to the patient condition and available data, if it is possible, the system makes a recommendation for the patient, if not, patient should connect to an online doctor or an expert who can provide them with personalized assistance. This model helps a patients prevent an urgent situation [11][12] [13].

3.2 Platform as a Service (PaaS):

In PaaS layer, the customer has possible control over the design, building, testing, updating and development of the online healthcare applications. So this service is used and seen by engineers. As Figure 3 shows, the chronic patient needs to receive a recommendation. The model for sending the recommendation is designed in PaaS model by use of a fuzzy logic system.

3.2.1 Fuzzy logic system and heart failure classification

Usually, in the domain of a health care system, the clinical data value is uncertain, dynamic, sophisticated and express in natural language. So there is no precise and direct translation from human language to computer language. That is why we use the fuzzy logic system to deal with the uncertainty in our clinical data set value for designing our online application in Paas model.
As we mentioned before, we start our study from heart failure disease.
Doctors usually classify patients' heart failure according to the severity of their symptoms into four classes. It places patients in one of four categories based on how much they are limited by physical activity. This classification is shown in Figure 4. Figure 5 shows the progression of a heart failure patient in the simulation graph. Any changes in the signs and symptoms can have an effect on the heart failure categories and change it from one class to another. In the progression of acute heart failure in Figure 6, A shows a good recovery after the first acute episode followed by a stable period. B shows that the first episode is not survived. C shows poor recovery followed by deterioration. D shows ongoing deterioration with intermittent acute episodes and an unpredictable death point. The significant heart failure signs and symptoms can be measured by the patient remotely and be sent to the clinical care department. These include:
edema, obesity, heart rate, heartbeat, blood pressure, saturation of oxygen and body temperature.
The clinical sign, demography, patient history etc can determine the class of patient's disease and based on this class, the patient can receive the suitable recommendation and treatment.
We will have a huge amount of data on the health variables that should be formulated to produce the output. [14]. The fuzzy logic system is an approach for computing based on a non-numeric value form of data.

4 Fuzzy logic process

Each Fuzzy system design includes the determination of input variables, output variables and fuzzy inference, which is a primary application of fuzzy logic. In proposed model, input variables are heart failure sign variables and output variables are classes of heart failure. Figure 7 shows inputs, outputs and the model for heart failure classification by use of a fuzzy logic system.

4.1 Input/output variables

Clinical variables are our inputs and class of heart failure is output. Membership function for input and output is defined from crisp set to present the degree of truth. We get clinical variables as input for classifying the heart failure disease into different stages. By using these inputs and their range, we can design membership functions of input variables. With Formula 1, for each language expression, we can obtain its membership as follows. Figure 8 shows the membership function chart the variable. The output is the class of chronic disease. These outputs also have to be defined with the Formula 1, so we consider a different output variable which is divided into the fuzzy set (State1, State2, State3, and State4).

$$\mu_{Low}(x) = \begin{cases} 1 & \text{x} < \text{a} \\ \frac{b-x}{b-a} & \text{a} \leq x < b \end{cases}$$

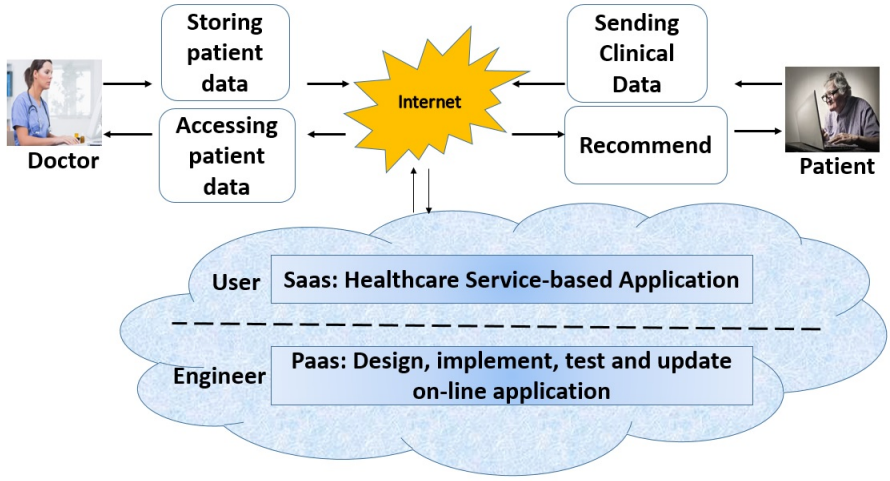


Figure 2: E-health online Algorithm

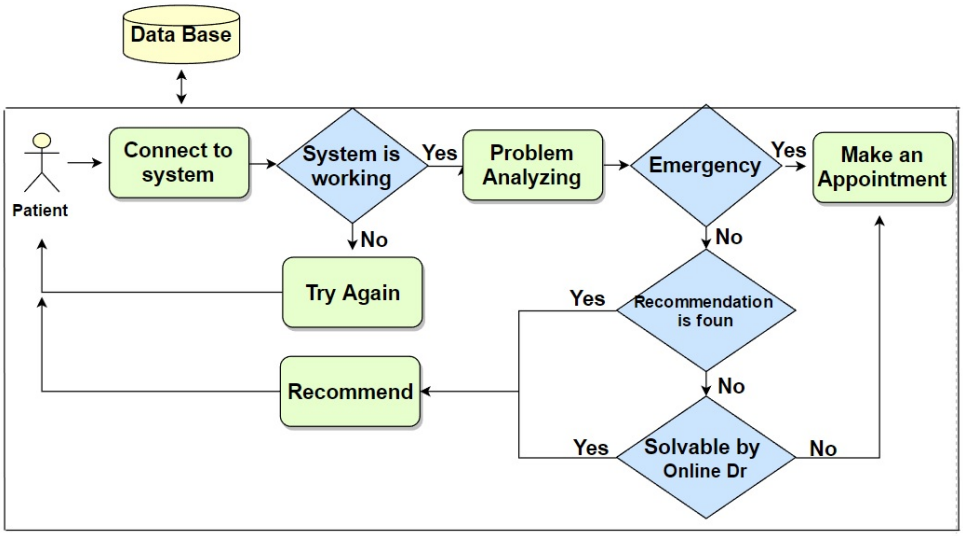


Figure 3: Finding recommendation model

$$\mu_{Mid}(x) = \begin{cases} \frac{x-a}{b-a} & \text{a} \leq x < b \\ 1 & \text{b} \leq x \leq c \\ \frac{d-x}{d-c} & \text{c} \leq x < d \end{cases}$$

$$\mu_{High}(x) = \begin{cases} (\frac{x-c}{d-c} & \text{c} \leq x < d \\ 1 & \text{x} \geq d) \end{cases} \quad (1)$$

4.2 Fuzzy Rules Base

After defining the membership function, we have to design the rule-base composed of expert IF-THEN rules[15]. Some parts of these rules are shown in Figure 9. These rules are driven by a different combination of clinical variables[16]. For example:
Rule 1: IF OEDEMA =Yes and Obesity= Yes and Heart Rate = High and Blood Pressure = High and Saturation of Oxygen =High and Temperature = High and Previos class = Class 4 THEN the State of Patient = Class 4.

4.3 Defuzzification

This step is the process that maps a fuzzy set to a crisp set. The proposed model can use the inference system whose output membership function is a fuzzy set. There are some different methods for the defuzzification whose center of gravity is most prevalent in defuzzification technique. This crisp set is an integer number. Formula 2 shows the method's center of gravity[16],[17] as follows:

$$D^* = \frac{\int D.\mu_M(D)dD}{\int \mu_M(D)dD} \quad (2)$$

NYHA Class	Level of Clinical Impairment
I	No limitation of physical activity. Ordinary physical activity does not cause undue breathlessness, fatigue, or palpitations.
II	Slight limitation of physical activity. Comfortable at rest, but ordinary physical activity results in undue breathlessness, fatigue, or palpitations.
III	Marked limitation of physical activity. Comfortable at rest, but less than ordinary physical activity results in undue breathlessness, fatigue, or palpitations.
IV	Unable to carry on any physical activity without discomfort. Symptoms at rest can be present. If any physical activity is undertaken, discomfort is increased.

Figure 4: New York Heart Association (NYHA) Functional Classification of Breathlessness, used in patients with heart failure, grading I-IV.

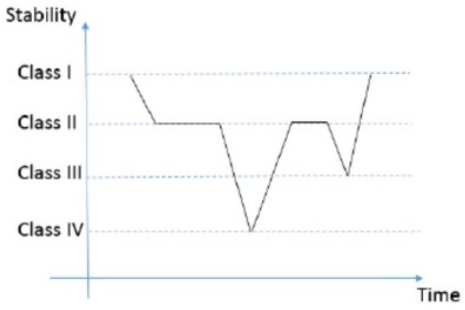


Figure 5: Heart failure classification

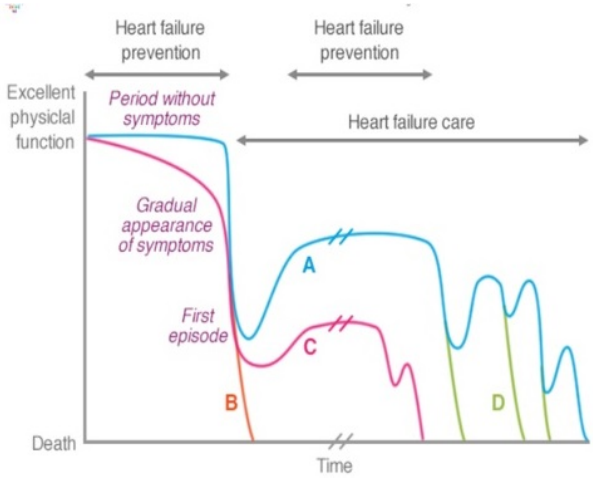


Figure 6: Heart failure classification

5 System implementation

5.1 The steps in the design of the E-health system

Figure 10 express the steps in design of the e-health model. First step is to design the patient model. In these steps based on clinical value and patient history, the patientlevel of danger is determined and then, in next step, the future situation of the patient is predicted. Finally, the recommendation is sent to the patient and the health care system is alerted.

5.2 Integrated E-health Cloud model

As we can see in figure 11, the clinical data of the patient is measured by the patient remotely and sent to the system by using a cloud service (SaaS) over the internet. This data is sent through Saas layer to Paas layer by use of smart devices. The Paas layer applies a fuzzy logic system and this data is used as an input for the fuzzy logic system. The output of the fuzzy logic system shows in which class a patient is placed and according to the class, a suitable recommendation is sent to the patient or the patient is connected to online doctor. The doctor or healh care staff are able to access this data and give treatment advice and update the patient's last situation.

6 Conclusions

Cloud computing can utilize the healthcare system by providing online monitoring services, application and keeping the data in storage. This data is proposed for analyzing by using fuzzy logic, classify the patient. Based on this classification, the recommendations is sent to the patients and alerts are created to the care system. It provides a better quality of service for health care system and a better quality of life for patient. It is helpful for the healthcare system management to have the possibility of predicting the critical time and situation of the hospital by using simulation. This proposed model has the possibility of being extended to each type of chronic disease and also being extended to more health variables as well as living and medical conditions. The designed model can be redesigned as

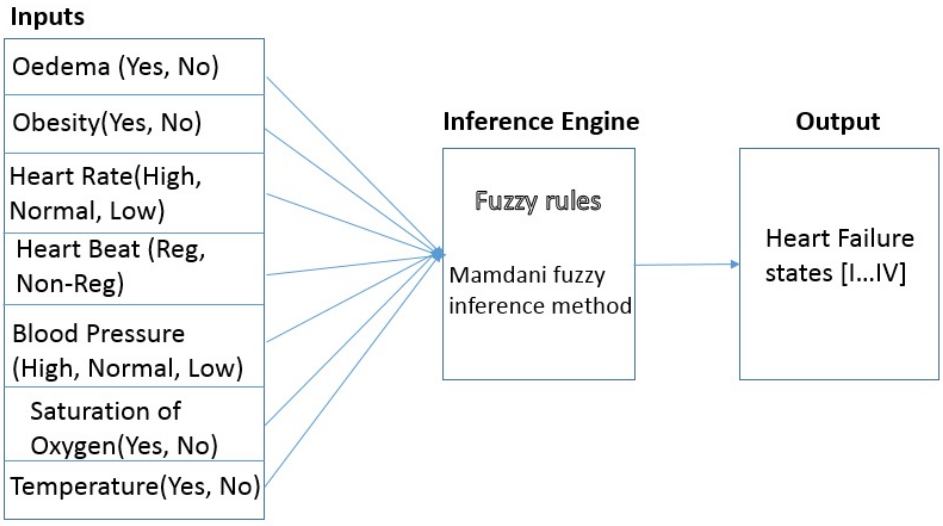


Figure 7: heart failure classification based on clinical variables

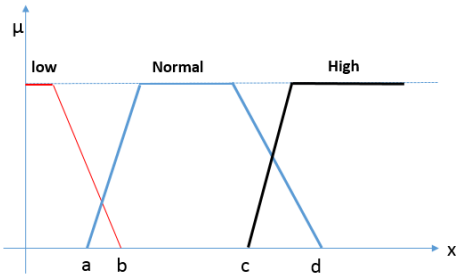


Figure 8: Member ship graphic

long as the analysis of simulation outcome shows that the result is precise and rational enough for the real world.

Competing interests

The authors have declared that no competing interests exist.

Acknowledgment

This research is supported by the MICINN/MINECO Spain under contracts TIN2014-53172-P and TIN2017-84875-P.

References

[1] H. Lewy, "Wearable devices-from healthy lifestyle to active ageing," in *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*, pp. 7748–7751, IEEE, 2015.

[2] E. Shojaei, E. Luque, D. Rexachs, and F. Epelde, "Simulation as a tool for evaluating intelligent self-care systems managing chronic disease patients," *International Journal of Integrated Care*, vol. 17, no. 5, 2017.

[3] C. Lisetti, F. Nasoz, C. LeRouge, O. Ozyer, and K. Alvarez, "Developing multimodal intelligent affective interfaces for tele-home health care," *International Journal of Human-Computer Studies*, vol. 59, no. 1, pp. 245–255, 2003.

[4] B. G. Celler, N. H. Lovell, J. Basilakis, *et al.*, "Using information technology to improve the management of chronic disease," *Medical Journal of Australia*, vol. 179, no. 5, pp. 242–246, 2003.

[5] F. Koehler, S. Winkler, M. Schieber, U. Sechtem, K. Stangl, M. Böhm, H. Boll, G. Baumann, M. Honold, K. Koehler, *et al.*, "Impact of remote telemedical management on mortality and hospitalizations in ambulatory patients with chronic heart failure: the telemedical interventional monitoring in heart failure study," *Circulation*, pp. CIRCULATIONAHA–111, 2011.

[6] Z. Liu, E. Cabrera, M. Taboada, F. Epelde, D. Rexachs, and E. Luque, "Quantitative evaluation of decision effects in the management of emergency department problems," *Procedia Computer Science*, vol. 51, pp. 433–442, 2015.

[7] S. H. Jacobson, S. N. Hall, and J. R. Swisher, "Discrete-event simulation of health care systems," in *Patient flow: Reducing delay in healthcare delivery*, pp. 211–252, Springer, 2006.

[8] M. Hamdaqa, T. Livogiannis, and L. Tahvildari, "A reference model for developing cloud applications.," in *CLOSER*, pp. 98–103, 2011.

Rule No	OEDEMA	Obesity	Heart Rate	Heart Beat	Blood Pressure	Saturation of Oxygen	Temperature
Rule1	Yes	Yes	High	Reg	High	High	High
...							
Rule38	Yes	Yes	Normal	Non-Reg	High	High	High
...							
Rule289	No	No	Low	Non-Reg	Low	Normal	Normal

Figure 9: Expressing the IF-THEN rules

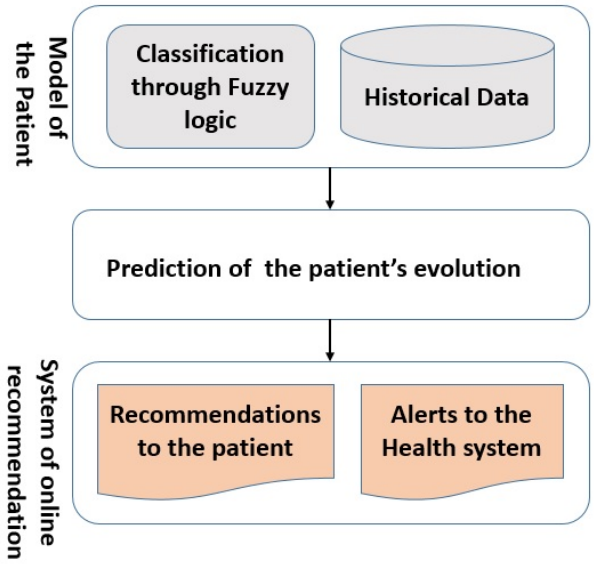


Figure 10: The steps in the design of the E-health model

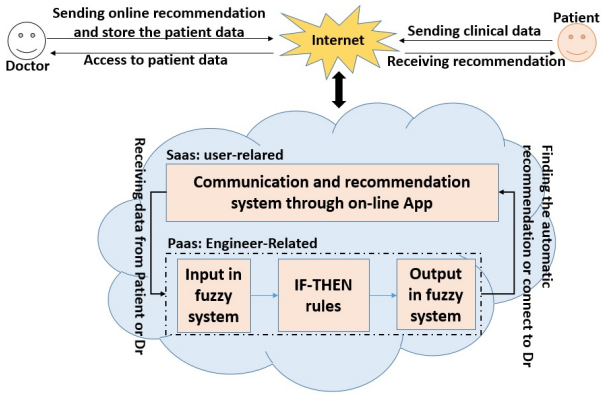


Figure 11: Integrated E-health model using cloud

[9] E. AbuKhousa, N. Mohamed, and J. Al-Jaroodi, "e-health cloud: opportunities and challenges," *Future Internet*, vol. 4, no. 3, pp. 621–645, 2012.

[10] T. Gao, D. Greenspan, M. Welsh, R. R. Juang, and A. Alm, "Vital signs monitoring and patient tracking over a wireless network," in *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*, pp. 102–105, IEEE, 2006.

[11] IBM, *Data-driven healthcare organizations use big data analytics for big gains*, 2013. Available via `http://http://www-03.ibm.com/industries/ca/en/healthcare/documents/Data_driven_healthcare_organizations_use_big_data_analytics_for_big_gains.pdf`.

[12] R. Reddy, K. Raju, M. Kumar, C. Sujatha, and P. Prakash, *Prediction of Heart Disease Using Decision Tree Approach*, 2017. Available via `http://www.ijarcsse.com/docs/papers/Volume_6/3_March2016/V6I3-0318.pdf`.

[13] A. Priyadarshi, *Decision Support System on Prediction of Heart Disease Using Data Mining Techniques*. International Journal of Engineering Research and General Science, 2015. Available via `http://pnrsolution.org/Datacenter/Vol3/Issue2/200.pdf`.

[14] O. Wolkenhauer, "A course in fuzzy systems and control," *International Journal of Electrical Engineering Education*, vol. 34, no. 3, p. 282, 1997.

[15] N. Allahverdi, S. Torun, and I. Saritas, "Design of a fuzzy expert system for determination of coronary heart disease risk," in *Proceedings of the 2007 international conference on Computer systems and technologies*, p. 36, ACM, 2007.

[16] I. Saritas, N. Allahverdi, and I. U. Sert, "A fuzzy expert system design for diagnosis of prostate cancer," *a a*, vol. 1, p. 50, 2003.

[17] C. Wang, *A study of membership functions on mamdani-type fuzzy inference system for industrial decision-making*. Lehigh University, 2015.

Fire propagation visualization in real time

Sigfrido Waidelich[1], Karina Laneri[2,3], and Mónica Denham[1,2]

[1]*Laboratorio de Procesamiento de Señales Aplicadas y Computación de Alto Rendimiento. Sede Andina, Universidad Nacional de Río Negro, Argentina*
[2]*Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina*
[3]*Física Estadística e Interdisciplinaria, Centro Atómico Bariloche, Río Negro, Argentina*
sigfri2wai@gmail.com, laneri@cab.cnea.gov.ar, mdenham@unrn.edu.ar

Abstract

Our motivation comes from the need of a tailored computational tool for simulation and prediction of forest fire propagation, to be used by firefighters in Patagonia, Argentina. Based on previous works on Graphic Processing Units (GPU) for fitting and simulating fires in our region, we developed a visualization interface for real time computing, simulation and prediction of fire propagation. We have the possibility of changing the ensemble of raster maps layers to change the region in which fire will propagate. The visualization platform runs on GPUs and the user can rotate and zoom the landscape to select the optimal view of fire propagation. Opacity of different layers can be regulated by the user, allowing to see fire propagation at the same time that underlying vegetation, wind direction and intensity. The ignition point can also be selected by the user, and firebreaks can be plotted while simulation is going on. After the performance of a high number of stochastic simulations in parallel in GPUs, the application shows a map of the final fire surface colored according to the probability that a given cell burns. In this way the user can visually identify the most critical direction for fire propagation, a useful information to stop fire optimizing resources, which is specially important when they are scarce like is the case of our Patagonia region.

Keywords: Forest Fire Simulation, GPGPU, High Performance Computing

1 Introduction

Every year, Argentina is the scenery of several and sometimes huge forest fires. Our country has a big extension of wild vegetation and firefighters unfortunately don't have reliable computational tools for management purposes.

In addition, our region still doesn't have fuel models for local vegetation to use as input for propagation and instead similar vegetation types from countries like Canada, Australia and USA are used [1, 2, 3].

Our team is working in collaboration with the fire brigade of "*Fire, Communications and Emergency Department*" (ICE) of the Nahuel Huapi National Park, in San Carlos de Bariloche. When a fire occurs, they mainly use their experience in the field to build the strategy to stop the fire. In some of the cases they used the outcome of FARSITE [4] but they noticed that simulations were very inaccurate depending on the vegetation that burns. Therefore we based our development on the needs of the ICE, a simulation tool adapted to our region with a friendly interface to be used to have more information to make decisions when fire is occurring.

Several simulators were developed to reproduce forest fire behavior, which need accurate inputs to give high quality results. Typical inputs are: topography (terrain elevation, slope and aspect), type of fuel, wind intensity and direction, and in several cases information from different weather stations. Simulators can be classified in to mainly two types: vector based and raster based [5]. Vector based approaches rely on elliptical waves propagation, they are more accurate for reproducing fire perimeter but they have greater time requirements. The most popular of this type of simulators is FARSITE [4]. Raster based approaches are based on a uniform grid in which fire propagates according to certain rules.

Some authors ([6], [7]) claim that the raster approach (including Cellular Automaton (CA) model) is more efficient than the vectorial approach. Vector implementation treats the fire perimeter as a closed curve, discretized through a number of points, each point spreads fire according its local conditions and fire propagation model. Then, the new fire perimeter is formed by the union of the outer shape of all individual fires. On the other hand the raster approach spread fire over a mesh of contiguous cells that can be inactive (not burning or burnt) or active (burning). Each cell has its own state and fire spreads from a cell to its neighbours based on a set of rules. It is interesting to note that raster approach matches with GPU execution model that uses a large amount of threads in a grid of threads in a SIMD (Single Instruction Multiple Data) model.

The use of GPUs shortened tremendously simula-

tion times but also offers the capability of rendering the simulation outputs in simulation time. In this sense they are optimal to visualize fire spread in reasonable times, which allow the use of these tools for decision making and management purposes.

Visualization of fire spread offers a number of benefits, that includes the comparison of model output with fires occurred in the past, but also the evaluation of the consequences of implementing preventing measures like firebreaks, prescribed burns or other changes in the environment. Using visualization is also possible to train firefighters and communicate results to the community in an easier way. If the visualization is also interactive it allows to test fire behavior after thinking strategies that could be very dangerous to test directly in the field. FARSITE has a visual interface that shows the simulation in 2D [4] but more attractive and interactive spatial visualization tools were built in GPUs [5].

Fire propagation models, on one extreme, can rely only in physical principles, in which case we have the advantage of understand the mechanisms but the disadvantage that very detailed and difficult to obtain input information is needed [5]. On the other extreme there are pure statistical models based on datasets to predict fire propagation for a particular situation but these models are very specific. And in between both approaches we can design semi-physical models that rely in some physical principles but are fitted to some data sets that allow to validate the model [5].

In previous works [8], a High Performance Computing forest fire simulator was developed as a parallel cellular automata implemented in CUDA-C ([9, 10, 11]). Our simulator spreads the fire over a landscape taking into account several layers of topography, vegetation type and wind ([12, 8]). The model for fire propagation is a semi-physical model and takes into account as input the actual vegetation, slope, average wind intensity and direction. The simulator is calibrated for a given region comparing thousand of simulations with the real fire final perimeter. The search in the parameter space was done using a Genetic Algorithm (GA), a previously implemented methodology [13, 14, 15] that we programmed in parallel using CUDA-C. A Monte Carlo method was also used to find the best parameters but GA was proved to be more efficient. The results of this calibration step are exposed in [8].

We now present the development of a visualization tool using OpenGL ([16, 17]), built on top of the simulator. Once the calibration was performed and the propagation parameters were estimated, we can simulate and visualize 10 stochastic simulations in parallel in the order of fraction of seconds which is acceptable for the time needed to make decisions in real time [18]. In this contribution we will not discuss in detail the model fitting stage that was previously explained elsewhere [8]. The main contribution of this work is the design and implementation of a powerful visualization tool for fire propagation, based on the needs of firefighters in Patagonia, Argentina.

2 Forest fire spread model

A Cellular Automaton was developed to simulate fire spread. The landscape is represented as a grid of cells and fire spreads over this grid. Each cell has its own state: burnt, unburnt and burning (cells that can spread fire to their neighborhood). Fire spreads to neighboring cells according to the following probability:

$$p = \frac{1}{1+\exp(-(\beta_0+\beta_1 I_f+\beta_2\psi+\beta_3\omega+\beta_4\sigma))} \quad (1)$$

Where I_f , ψ, ω and σ, are related with the vegetation type, aspect, wind direction and slope respectively, as explained in [8, 12]. Fuel type coefficient β_0 is the baseline fire propagation probability for shrubland cells and β_1 is the difference between shrubland and forest. Parameters β_2, β_3 and β_4 modify propagation probabilities according to aspect (ψ), wind direction (ω), and slope (σ) respectively [12].

In order to determine if a target cell is reached by fire, the ignition probability is calculated taking into account the state of the 8-cells neighborhood using Equation 1 as explained in [8]. The following pseudocodes illustrate the main characteristics of algorithms (in order to improve clarity these algorithms does not show visual interface details):

Algorithm 1 Main loop

```
1: init all maps                          ▷ Read maps from hard disk
2: init input simulator parameters        ▷ β_i values
3: while fire spreads do                  ▷ CA simulation steps
4:     parallel CUDA thread grid execution
5:     fire map update
6:     fire spread flag setting           ▷ If fire stops
7: end while
```

Algorithm 1 presents the main fire spread simulation loop, which is executed to perform a complete simulation. Firstly, input simulator parameters are initialized (Algorithm 1 lines 1 and 2). Then, while fire spreads, the loop block is executed: a CUDA thread grid is launched to perform each CA step. The thread grid is a 2D matrix of threads, with the same dimension of the raster maps. Then, one thread per map cell is launched and all cells of the new fire map are calculated concurrently (or parallel execution depending on the number of CUDA cores available on the graphic card). All threads execute the same function (CUDA kernel) in a SIMD way. In a more accurate analysis, when CUDA application is analyzed the model becomes to SIMT: Single Instruction Multiple Thread since each thread can take its own instruction trace.

Once the kernel function ends, the fire progress map is updated with the new fire state. Furthermore, if

no cell changes to the burning state, then fire does not spread (if borders are reached or the fire can't propagate any more), and a flag is set *false* to stop the simulation.

Algorithm 1 presents a complete simulation, that is one of the options of our simulator. When user needs, the CA advances just 10 time steps. Section 4.2 will present this propagation possibility.

Algorithm 2 Thread$_{i,j}$ algorithm

```
1: thread position ← grid row i and column j
2: if cell_{i,j} is burnable then
3:     for each neighbor k of cell_{i,j} do
4:         PoI = f(p_k)
5:     end for
6:     if PoI > random value then
7:         cell_{i,j} = burning
8:     end if
9: end if
```

The Algorithm 2 illustrates the inner loop that is executed to spread the fire (line 4 Algorithm 1). First of all we verify that the target cell is burnable (Algorithm 2 line 2) and if this is the case, we proceed to evaluate the Probability of Ignition (*PoI* in line 4 of Algorithm 2) of the target cell produced by at least one of its neighbors. This probability of ignition (*PoI*) is computed evaluating Equation 1 for each of the k neighbors and combining all the contributions in the function represented by f in line 4 Algorithm 2, as explained in [8].

The target cell will be burned according to that probability (*PoI*) by throwing a random number as in line 6 of Algorithm 2. All is done in parallel and once all burning probabilities are computed by each of the threads the whole fire map is updated at once.

The above explained simulation procedure is done millions of times to fit the parameters of the model to the real fire data. This fitting step is done in order to find the best set of simulator parameters (β_i values in Equation 1) [8].

The fitting procedure is done by minimizing the distance, or fitness, between real fire and simulation. The fitness is computed as the number of burned cells that have in common both, simulation and real fire. Fitness equal to zero indicates that the fire after simulation is identical to the real fire. The search of the ensemble of parameters β_i that best fit the model to data, was done using a Genetic Algorithm (GA). This search strategy consists in changing randomly some β_i, perform a simulation and compute the corresponding fitness. The β_i values are changed by performing selection, crossover and mutation between different ensembles, after every change a simulation is performed and the fitness of the simulation is calculated comparing with real data.

In addition to GA, a Monte Carlo method was implemented, where β_i values where chosen from the valid range of parameter values simulating a brute force method. The GA converges quickly to good results [8].

The fitness values are ranked in decreasing order and the corresponding β_i are therefore ranked together with their fitnesses. A histogram of the best ranked individuals allows to determine the best values of β_i and the associated errors, as explained in more detail in [8].

After the fitting procedure we obtain a best set of β_i parameters for each of the Patagonia regions presented, i. e. *Falso granítico* and *Laguna Seca*. Is with those parameters that we will simulate and visualize several simulations to have a measure of fire extension and propagation. As the simulations are stochastic, the use of the best set of parameters will give rise to different fire scars.

3 Simulator inputs and outputs

Simulator inputs are several maps that describe the landscape (terrain vegetation, slope, aspect and elevation), as well as wind speed and direction.

Fire environment is described by several raster maps that represent the area of interest as a grid of cells. Each raster map has the description of the area that represents: number of rows and columns, cell sizes, and coordinates of one corner. Using this information, each map can be georeferenced using GIS software (including Google Earth).

For topography description DEM (Digital Elevation Model) maps were used. Nowadays several satellite and radar information is available from different data sets on Internet. For example, USGS Earth Explorer offers maps of several sources and types: LandSat, Sentinel, ASTER global DEM collections for example, with high temporal and spatial resolution [19]. This information is freely available and we are using DEM raster maps from this site. Therefore using GIS tools, slope and aspect maps can be obtained from the elevation raster map. The elevation map is used to visualize the topology of the area of interest. Slope and aspect maps are inputs for the fire spread model.

Fuel type information, as described on [1, 2, 3], are specific combinations of vegetation types that together define the fire behavior. For Argentina, fuel types are still not characterized as in [1] and [2]. As a surrogate of fuel type we will use vegetation type. Combining vegetation type with fire behavior is a very important task to be accomplished, because fuel type often determines fire behavior (specially when nor slope neither wind are strong enough to drive fire propagation).

Our simulator uses wind velocity and direction for fire spread computation. However for the moment we are using average quantities and it would be a future task to include wind variability and ultimately coupled wind models [18] to allow a more accurate real-time prediction of fire propagation.

At the moment, we are working with 2 real fires

occurred at the Northwestern Patagonia Andean region (Figure 1). This Figure shows estimated ignition points of both real fires. The first test case is called *Falso Granítico* fire, that occurred in 1999, in the *Falso Granítico* Hill proximity (at 41°21′59″ S, 71°38′46″ O). Raster maps of 801x801 cells were used for this case. The second test case corresponds to a fire occurred in 2012 near to *Laguna Seca* (at 41°02′35″ S, 71°16′36″ O). Raster maps of 181x423 cells were used for this case. We constructed slope and aspect raster maps from DEM information. In both cases, map resolution is 30x30m.

Figure 1 shows different vegetation types. More detailed vegetation cover classification for Patagonia Andean region is available in [20] (updated in 2016).

In Figure 1 vegetation classification corresponds to: forest type A where predominates native forest (lenga, coihue, cypress), forest type B where predominates low vegetation (lenga, ñire, and mixed woods) and forest type I to indicate exotic forest. Grassland includes steppe, marshland and wetland. Finally, shrubland includes shrubland (native and exotic) and infrastructure areas (to avoid more vegetation type divisions). No fuel includes rocks, lakes, bare ground, ice and snow.

In order to feed our simulator we transform this data to three vegetation types. At the moment our simulator considers three fuel types: shrub, forest and no fuel. Including more accurate and real vegetation types is one of the most important open lines in a near future.

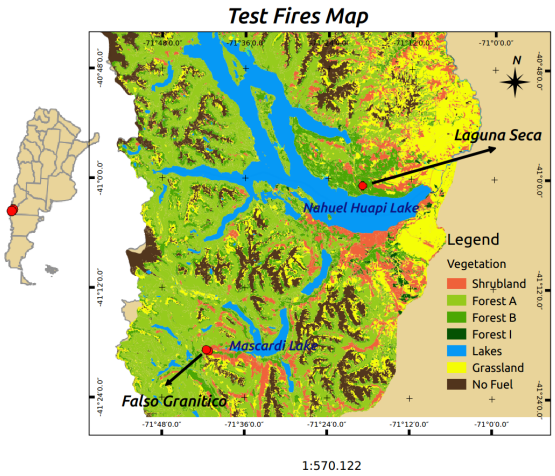


Figure 1: Area of interest situated at the Nahuel Huapi National Park in the North West Patagonia. Arrows indicate the two ignition points of *Laguna Seca* and *Falso Granítico* fires.

The simulator output is a raster map where each cell is labeled according to the resulting probability of ignition after the execution of 10 simulations (this number can be changed by the user and is limited by the memory of the graphic card). The output map can be georeferenced using GIS, given that meta data is copied from the input raster maps.

4 Forest fire simulator

Our application has the ability to perform high performance simulations of the spread of wild fire in a very complex environment. In addition it incorporates a powerful visualization platform, including an intuitive interface for user interaction, to command fire progress.

Most of the times, the visualization of a problem helps to understand the behavior of a complex phenomena. Our simulator shows fire progress over the landscape, where the action of topography, vegetation and wind can be tested and modify. For instance, we provide a helpful user interface, where firebreaks can be easily set, fire can move forward and backward, the ignition point can be changed by the user, etc.

OpenGL ([16, 17]) was chosen to program the simulator visualization and user interaction. Therefore, fire environment visualization and fire progress are rendered by graphic cards, as well as user interaction (with mouse and keyboard). Finally as previously mentioned, CA for fire spread simulation is also executed in graphic cards. In summary, simulation fire progress, visualization in real time and user interaction are all solved by GPUs, that deal with all these high performance requirements in a successful way.

Next paragraphs will expose visualization and afterwards the most relevant tools of our simulator. We will explain important simulator characteristics, as well as design details and some implementation decisions.

4.1 Forest fire progress visualization

Figures 2 and 4 show our application main window. The left panel shows the fire progress and the ignition probability of cells. The ignition point was arbitrarily set by the user. Ten simulations are performed in parallel starting from exactly the same ignition point. Given that the model is stochastic every realization is different from the previous one. While performing 10 simulations in parallel the burning probability of each cell is determined. If a given cell was burnt in all of the simulations, the burning probability is one, but if only some of the times the cell with coordinates (i, j) burns, the probability will be:

$$bp_{i,j} = \sum_{1}^{10} \frac{Times\ burns}{10} \qquad (2)$$

According to Equation 2 if a given cell burns in all the simulations then bp=1. We colored the final simulation according to this probability, being red for bp=1 and more yellowish when this probability diminishes.

When the mouse pointer moves over the map cells, a label with the ignition probability information is shown. This probability is the one displayed in Equation 2, i. e. the number of times that this cell burned through the 10 simulations. As the model is stochastic, the simulations are different from each other. Figure 3

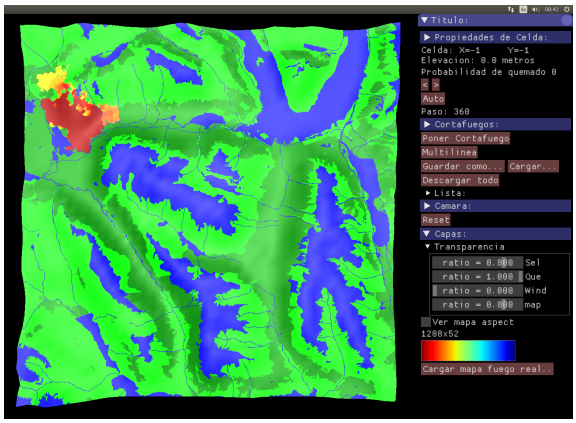


Figure 2: Top view of the landscape: burnable cells (green) correspond to different types of vegetation, unburnable cells (blue) are lakes, rocks, etc. Fire is showed in red (high burning probability) and yellow (lower burning probability).

shows the same simulator top view where the mouse pointer is labeled with cell fire probability. This Figure includes the wind layer (arrows above the terrain). This simulator feature will be explained later.

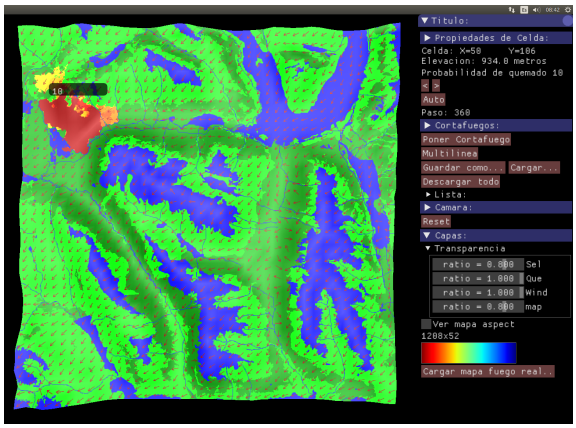


Figure 3: Top view of the landscape: arrows represent average wind direction and arrow colors represent wind intensity. More wind (red) and less wind (blue). Fire scar is shown in red/orange/yellow according to the probability that cell burns after ten simulations. Higher probability (red) lower probability (yellow).

The number of parallel simulations to be performed can be changed. Input maps are loaded once in GPU global memory. These maps are read by kernel threads through each of the simulations. However each of the parallel simulations need current fire map and a new fire map for labeling each cell that is reached by fire. Therefore, these two maps (with fire spread state) are allocated for each of the 10 simulations since vegetation, slope, aspect, wind maps are allocated just once (they are read only input maps). Our application takes care of the use of GPU memory.

The visualization of fire progress is updated every 10 cellular automaton steps. This value can be changed, but we could see that this value is useful for clarity. Furthermore, every 10 steps fire maps are saved in order to go back and reproduce past states of the fire. For improving application efficiency just the coordinates of the ignited cells are stored. In this way past fires can be easily recovered. Then, fire can advance or rewind as the user needs for simulation observation. This simulator feature in combination with firebreak setting are very important and powerful tools for firefighters and other users. Two buttons on the main menu and two keys allow to command fire progress forward and backward.

Map view (left panel on the simulator Figures) can be changed as user need: top view is the default view (Figure 2 and Figure 3), then user can zoom in and out the map, and drag it to view adjacent areas. Furthermore, the map can also be rotated to modify this view.

This actions are performed on the graphic card. Figure 5 and Figure 6 show different views of the same map (*Falso Granítico* fire area). Figures5 and Figure 6 show the left top map corner zoomed in and rotated. Terrain elevation is accentuated in these last figures. Each time, the default top view can be set with the Camera Reset button.

The performance of visualization and user interaction is excellent. For example due to design and implementation characteristics, setting firebreaks and changing map view (zoom, drag and rotation) needs lot of computation for updating fire map. We can see that GPUs solve this requirements in a very successful way. We tested our simulator in different computers and graphic cards and it executes with no delays both computation and interaction.

Next paragraphs present the simulator tools implemented until now and in the section open lines we will explain the additional functionality that is planned to be added to our simulator.

4.2 Interaction tools

Most of the simulator options and functionality is controlled with the right panel menu. Figure 4 shows this main panel with the most important interaction tools classified in different sections.

First menu section shows cell features: the user can move the mouse pointer over the map and the most important cell characteristics are shown: cell coordinates, terrain elevation and burning probability (if cell was reached by fire).

Then, two buttons appear with $<$ and $>$ symbols. This buttons allow the user to select to go forward or backward the fire progress. This functionality is available with $\leftarrow$ and $\rightarrow$ keyboard buttons also. As mentioned, 10 cellular automaton steps are performed each time the $>$ or $\rightarrow$ buttons are pressed. The Auto button executes CA steps until the simulation is completed (process showed on Algorithm 1).

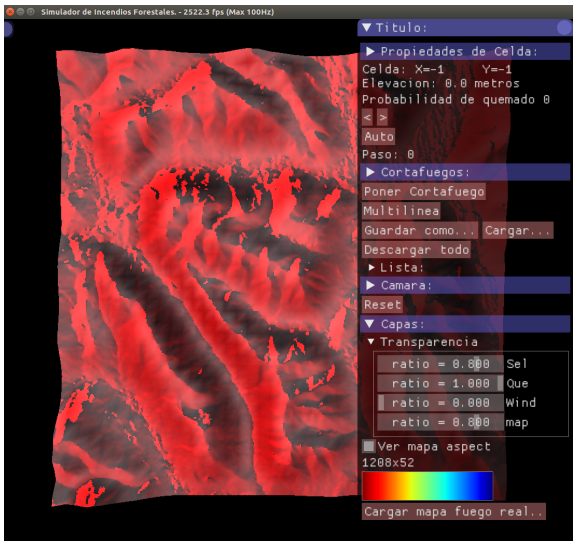


Figure 4: Simulator main menu. In this case the aspect map view is shown: this view emphasize slope orientation.

Next menu section defines firebreak functionality. This panel section allows the user to set firebreaks over the landscape. That means that these parts of the terrain are set as nonburnable cells. Once the firebreak button is selected, the mouse is used to draw the firewall.

The user can set line firewalls as well as multiline firewalls (freehand drawing). Figures 5 and 6 show the same time step of the CA without firebreak and when a firebreak was set respectively. When a firebreak is set, fire doesn't spread to the north-west (in this case) of the terrain.

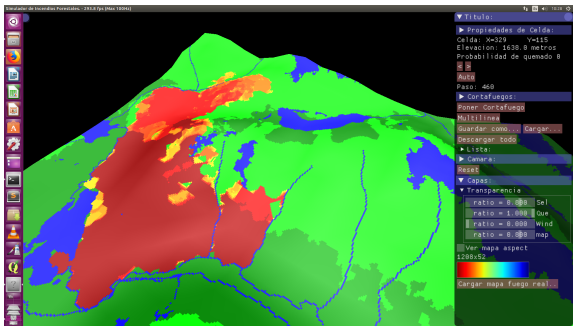


Figure 5: Top left corner of previous figure zoomed and rotated. Cellular Automaton was executed 460 times. No firebreak was set.

Firewalls can be stored in files and they can be loaded at any time. The combination of setting firewalls and choosing when fire goes forward or backward allows the user to understand where firewalls are more efficient, in combination with topography, vegetation, etc. This is a powerful capability of our simulator.

Firebreaks are limited by raster cells shape. Fire

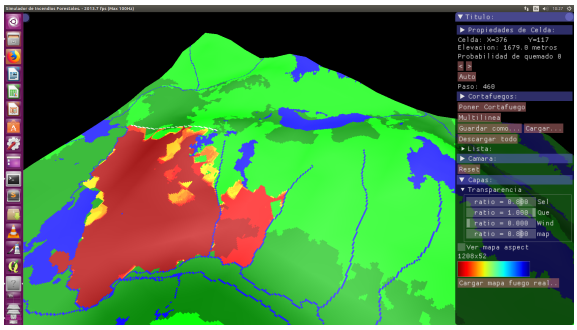


Figure 6: Same simulation step than the previous figure but with a firebreak (white line) at the top left corner. The fire does not spread to the north of the terrain. Cellular Automaton was executed 460 times.

propagation through a diagonal firebreak is avoided calculating when fire has to spread or not when target cell is adjacent to a firebreak. For example, authors of [5] mention the same problem and they decided to modify the thickness of the firebreak line. Firebreak thickness was set to twice the size of a single cell. Our approach maintains the thickness of firewall but calculating the interaction of the fire with the firebreaks diagonal cells.

The next section manages camera options. As already mentioned, fire area can be zoomed, rotated, relocated and simulator window can be reshaped. Camera Reset button allows the user to set the default top view (Figure 2).

The last menu section allow to manage several raster maps as information layers that can be visible or not, depending on their transparency. Wind layer, fire layer and firebreaks layer can be turned on and off by changing layer level of transparency. Using transparency sliders, the users can change which information to visualize, depending on their needs. If aspect terrain information is useful then the user can check this option (this check box is selected in Figure 4).

More layers are planned to be included. For example, wild fires final burnt area can be obtained from aerial images. Then, using GIS tools (digitalization process) we can obtain vectorial or raster maps with the final burned area. At the moment our team is working in order to include this final fire map as a simulator layer. As other layers, user will be able to set its transparency in order to visualize this map in combination with other layers. This will be a powerful tool for testing firebreaks and simulations accuracy.

Wind speed and direction is showed as a wind layer. User can manage layer transparency to see wind features. This information is presented by colored arrows. Each arrow shows the direction and intensity of wind.

Wind intensity is calculated using a color ramp: blue are minimum velocities and red are higher velocities. The color ramp is included in the main menu. Arrow color is calculated for each map using an equalized histogram for wind speeds. Therefore, less frequent

values are ignored and most frequent values are used for speed values discretization.

Wind direction is represented by arrows. Each arrow represents the average of a group of cells values. Depending on the map resolution, if each of the wind cell is represented by a unique arrow, the visualization gets confused specially when combined with other layers. Therefore, we choose to divide the number of cells proportionally to the map dimension size.

Border cells were a particular difficulty for visualization of wind arrows. As they have less than 8 neighbors, the average was calculated taking into account the available neighbors.

4.3 OpenGL programming

When the simulator is launched all fire environment maps are read from the hard disk. The OpenGL functionality is also initialized: several functions (callbacks) are registered to OpenGL in order to solve the simulator window display, keyboard button pressing (down or up), mouse motion (active or passive) and simulator window reshape.

To implement the simulator main menu, the library ImGUI (Immediate Modal Graphical User Interface [21]) was used. ImGUI is a graphical user interface library for C++. This library is particularly suited for integration in game engines (for tooling), real-time 3D applications, full screen applications, embedded applications, etc. This library offers the implementation of menu buttons, panels, lists, checkboxes, radio button, labels, sliders, etc.

The simulator main window shows fire progress map. This map is an OpenGL texture calculated from different bitmaps. Our simulator manages 4 bitmaps: vegetation map, fire breaks map and fire map. These bitmaps have the same fire maps dimensions (rows x columns). Each bitmap pixel corresponds to a map cell. Each pixel is a (r,g,b,alpha) tuple, where r, g and b form pixel color and alpha is the transparency of the pixel. Using these 3 bitmaps a final bitmap is formed (this is the texture that the simulator passes to OpenGL to draw each frame).

There are some events that cause final texture update. This texture update will be used during next window frames actualization. Different operations cause texture regeneration: fire spread changes (moving forward or backwards), line or multiline firebreak setting, saved firebreak loading, layer slider controls settings, etc.

When a texture update occurs, bitmaps are copied to GPU, then, a CUDA kernel is launched in order to compute final texture, pixel colors and transparencies, and this new texture is used to update the fire progress map on the following frames.

In a low level of implementation all the visualization is solved by OpenGL vertexes, triangles and indexes. Indexes are converted to bitmap UV coordinates to form textures. Camera and light position (ambiance light and spectral light), near and far planes, zooms, dragging and rotation of the figure are solved by low level matrices that multiply vertexes or are used to define colors and transparencies.

In order to calculate map zooms, dragging, rotations, etc, some transformation matrices are used and joined with the map model. For example, if user has zoomed in and rotated the image, 4x4 matrices for this operations are multiplied and the result is saved in the transformation matrix. Then, this transformation matrix is used to multiply each vertex to obtain the new vertex values.

These operations are optimized in GPGPU. The simulator response time and user interaction are solved in a very successful way in CUDA-C and OpenGl.

5 Conclusions and Open Lines

A visualization tool for fire propagation was developed in OpenGL on top of a forest fire simulator previously developed in CUDA-C. We presented a useful computational tool to be used for fire management, training and communication. It was developed on graphic processing units to meet the requirements of a high performance real time application, with a friendly graphical interface to be use for firefighters.

Models based more on the physical mechanisms of fire propagation, such as the physical-statistic model defined by Rothermel will be implemented in the near future. This will help to better understand the mechanisms involved in fire propagation in our region. To this aim it's necessary to characterize the fuel types for Patagonia, a task that should be in the agenda of natural resources management.

To improve prediction, it will be necessary to include the wind variability both in direction and intensity in real time. Another possibility will be to couple a wind simulator to the model simulator. This would be a very challenging task because the whole simulator will have to accomplish with the real time requirements.

Our model includes three vegetation types, but we have access to more detailed vegetation maps that should be conveniently transformed to raster maps, to be included in the model.

Another open line is to fit the advance of the fire front. In previous works we fitted our model to the final fire scar but intermediate fires scars can be obtained using satellite images or will be provided by our collaborators at ICE.

One of the useful features of our previous work was the possibility to determine the ignition point with its corresponding uncertainty. The ignition point could be caused by natural factors, like thunders or by anthropic causes which is mostly the case in our region. The possibility to visualize the ignition point if its unknown, will probably help to get clues about the fire origin.

Other feature to be included into the visualization interface will be a sensitivity analysis of the parameters.

Until now we are visualizing the output of 10 simulations performed with the best set of parameters. However with our fitting procedure we can determine an error associated with those parameters. Changing one of the parameters in its error range, but fixing the rest in the best values, it is possible to measure how sensible are the simulations of fire propagation to a change of that parameter. In this way we can add to the visualization panel interface, some slider bars representing the possible variability range of each parameter. With this tool, the user can easily explore the sensitivity of simulations when parameters are changed, generating less probable but also possible fire scenarios.

6 Acknowledgements

We acknowledge Marcelo Bari and collaborators from ICE (Departamento de Incendios, Comunicaciones y Emergencias del Parque Nacional Nahuel Huapi, APN: Administración de Parques Nacionales). For helpful comments on maps and GIS we thank Gabriela Denham. M. Denham and K. Laneri are members of Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina. M. Denham and K. Laneri are part of the project TIN2014-53234-C2-1-R of Ministerio de Ciencia e Innovación (MICINN-Spain). M. Denham, K. Laneri and S. Waidelich are part of the project PI-UNRN "Aplicación de Alto Rendimiento para Predicción del avance del fuego en Incendios Forestales".

References

[1] J. H. Scott and R. E. Burgan, "Standard fire behavior fuel models: A comprehensive set for use with Rothermel's surface fire spread model," tech. rep., United States Department of Agriculture. Forest Service. Rocky Mountain. Research Station, 2005.

[2] E. H. Anderson, "Aids to determining fuel models for estimating fire hehavior," tech. rep., United States Department of Agriculture. Forest Service, 1982.

[3] S. Taylor, R. G. Pike, and M. E. Alexander, *Field Guide to the Canadian Forest Behaviour Prediction (FBP) System*. Canadian Forest Service, 1996.

[4] M. A. Finney, "Farsite: Fire area simulator - model development and evaluation," tech. rep., USDA Forest Service, 2004.

[5] R. V. Hoang, M. R. Sgambati, T. J. Brown, D. S. Coming, and F. C. Harris, "VFire: Immersive wildfire simulation and visualization," *Computers & Graphics*, vol. 34, no. 6, pp. 655 – 664, 2010.

[6] T. Ghisu, B. Arca, G. Pellizzaro, and P. Duce, "A level-set algorithm for simulating wildfire spread," *CMES Computer Modeling in Engineering and Sciences*, vol. 102, no. 1, pp. 83 – 102, 2014.

[7] T. Ghisu, B. Arca, G. Pellizzaro, and P. Duce, "An improved cellular automata for wildfire spread," *Procedia Computer Science*, vol. 51, no. Supplement C, pp. 2287 – 2296, 2015. International Conference On Computational Science, ICCS 2015.

[8] M. Denham and K. Laneri, "Using efficient parallelization in graphic processing units to parameterize stochastic fire propagation models," *Journal of Computational Science*, vol. 25, pp. 76 – 88, 2018.

[9] T. M. John Cheng, Max Grossman, *Professional CUDA C Programming*. Wrox, 2014.

[10] S. Cook, *CUDA Programming. A developer's guide to parallel computing with GPUs*. Morgan Kaufmann Publishers Inc., 2013.

[11] D. B. Kirk and W.-m. W. Hwu, *Programming Massively Parallel Processors: A Hands-on Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1st ed., 2010.

[12] J. M. Morales, M. Mermoz, J. H. Gowda, and T. Kitzberger, "A stochastic fire spread model for north patagonia based on fire occurrence maps," *Ecological Modelling*, vol. 300, no. 0, pp. 73 – 80, 2015.

[13] M. Méndez Garabetti, G. Bianchini, M. L. Tardivo, P. Caymes Scutari, and G. V. Gil Costa, "Hybrid-parallel uncertainty reduction method applied to forest fire spread prediction," *Journal of Computer Science and Technology*, vol. 17, pp. p. 12–19, Apr. 2017.

[14] M. Denham, "Dynamic data driven application for forest fire spread prediction," *Journal of Computer Science and Technology*, vol. 12, pp. p. 84–86, Aug. 2012.

[15] M. Denham, *Predicción de la Evolución de los Incendios Forestales Guiada Dinámicamente por los Datos*. PhD thesis, Universidad Autónoma de Barcelona, 2009.

[16] E. Meiri, "OGL. Modern OpenGL Tutorials."

[17] E. Luten, "OpenGL Book."

[18] G. Sanjuan, T. Margalef, and A. Cortés, "Applying domain decomposition to wind field calculation," *Parallel Computing*, vol. 57, pp. 185 – 196, 2016.

[19] U.S. Department of the Interior, "USGS: U.S. Geological Survey. Earth Explorer."

[20] CIEFAP-MAyDS, "Actualización de la Clasificación de Tipos Forestales y Cobertura del Suelo de la Región Bosque Andino Patagónico. Informe Final," tech. rep., Centro de Investigación y Extensión Forestal Andino Patagónico, 2016.

[21] O. Cornut, "Immediate mode graphical user interface."

Job Schedulers for Machine Learning and Data Mining algorithms distributed in Hadoop

[1] Cornejo, Félix Martin; [2] Zunino, Alejandro; [3] Murazzo, María

1,3 Departamento de Informática, Universidad Nacional de San Juan;

2 Instituto Superior de Ingeniería de Software de Tandil, Universidad Nacional del centro de la provincia de Buenos Aires

1 fmartin.cornejo@gmail.com; 2 azunino@gmail.com; 3 maritemurazzo@gmail.com

Abstract [1]

The standard scheduler of Hadoop does not consider the characteristics of jobs such as computational demand, inputs / outputs, dependencies, location of the data, etc., which could be a valuable source to allocate resources to jobs in order to optimize their use. The objective of this research is to take advantage of this information for planning, limiting the scope to ML / DM algorithms, in order to improve the execution times with respect to existing schedulers. The aim is to improve Hadoop job schedulers, seeking to optimize the execution times of machine learning and data mining algorithms in Clusters.

Keywords: Big Data, Hadoop, schedulers of Hadoop, ML/DM algorithms, machine learning.

1. Introduction

Recently, the valuable knowledge that can be extracted from the analysis of sets of large-scale data, has given rise to the so-called "Big Data". The term "Big Data" [1,2] refers to a collection of large data sets that can not be processed using traditional database administration tools. This generated numerous scientific challenges such as how to provide support for storage, manipulate and then retrieve information. In this sense, it is necessary to have capable infrastructures to process large volumes of data, in acceptable times and with limited computational resources. Solutions typically rely on concepts of parallel and distributed computing, where through Clusters and Computational Grids [3] allow processing Big Data at relatively low costs [4,5].

Although the fact that parallel and distributed computing appears as one of the most interesting alternatives to store and manipulate Big Data, some of its characteristics prevent its use by part of common users [6]. Aspects such as data dependencies, integrity, load balancing, planning and fault tolerance, although extremely difficult toaverage programmers, are crucial for Big Data solutions to be operational. For this reason, several frameworks have been proposed that abstract these aspects and provide high-level solutions for users and programmers [5,6,7,8,9,10,11]. Some of those frameworks they are based on specific programming paradigms such as Fork-Join, MapReduce and MPI. Recently, the MapReduce paradigm attracted attention due to its applicability in parallel computing, being widely used by Google in its distributed file system GFS [12]. The great novelty of the paradigm and its implementation in a framework consists of in that it conceals to the programmer great part of the complexity of parallelization and distribution. Thus, programmers can focus on the functionality of their programs, while the framework abstracts the complexity and control the underlying computational infrastructure. One of the most successful implementations of MapReduce is Apache Hadoop In addition, it provides solutions to store, manipulate and extract Big Data information in different ways. Around Hadoop has flourished an ecosystem of solutions for specific problems. Some examples are Pig, which facilitates the analysis of large datasets, Spark that speeds up MapReduce using structures in RAM and Mahout that aims to scale data mining and machine learning algorithms.

While these efforts have had a significant impact on research and industrial environments the diversity of uses of Hadoop has led to underperforming results [13]. This is due, among other factors, to the scheduler, component responsible for assigning the

computational resources of a Cluster to the works, is one of the aspects that most influence the performance of Hadoop. Originally, Hadoop includes three schedulers: FIFO, Fair and Capacity. The default scheduler is FIFO, which queues jobs in order of arrival and executes them. The Fair scheduler, developed by Facebook, seeks to allocate equitable Cluster resources to tasks. The Capacity scheduler was developed by Yahoo! to ensure equitable allocation for a large number of Cluster users.

Although these Hadoop schedulers give flexibility so that users can optimize the execution of their works to the Cluster, the resulting performance is often less than expected [14] resulting in not only delays in execution time, but also low use of Cluster resources. Some efforts have proposed ad-hoc schedulers for Hadoop that consider different aspects of both tasks and computational resources available.

In this paper we analyze and discuss previous efforts on improving Hadoop schedulers and present our current research to improve Hadoop schedulers. The rest of this paper is organized as follows. In Section 2 we provide background about Hadoop and its built-in schedulers. Then, Section 3 overviews and discuss the most representative research on Hadoop scheduling. Section 4 proposes possible current and future research towards improving Hadoop scheduling for machine learning and data mining tasks.

2. Background

A significant amount of research has been done in the field of Hadoop Job scheduling; however, there is still a need for research to overcome some of the challenges regarding scheduling of jobs in Hadoop clusters. Industries estimate that 20% of the data is in structured form while the remaining 80% of data is in semi structured form. This is a big challenge not only for volume and variety but also for data processing, which can lead to problems for I/O processing and job scheduling. Fig. 1 shows the architecture of a Hadoop distributed system.

As we know, this is an era of Big Data where machines process a significant amount of data, in the range of terabytes or petabytes, using various applications in fields such as science, business, and commerce. Such applications require a considerable amount of I/O processing and spend most of the time doing I/O processing, which is a major part of job scheduling. It is reported that at the Facebook and Microsoft Bing data centers, I/O processing requires 79% of the jobs' duration and 69% of the resources. Therefore, here we present a comprehensive study of the most representative Hadoop schedulers.

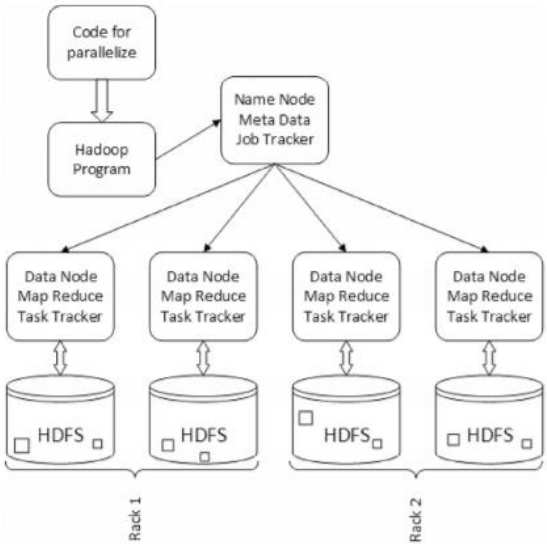


Fig 1. Hadoop distributed system architecture

2.1. Apache Hadoop Ecosystem

The Apache Hadoop is a framework that allows the processing of large volumes of data through Cluster systems, using a simple programming model. In addition, its design allows the scalability from a few nodes to thousands of nodes in an agile way, achieving high efficiency in vertical scaling.

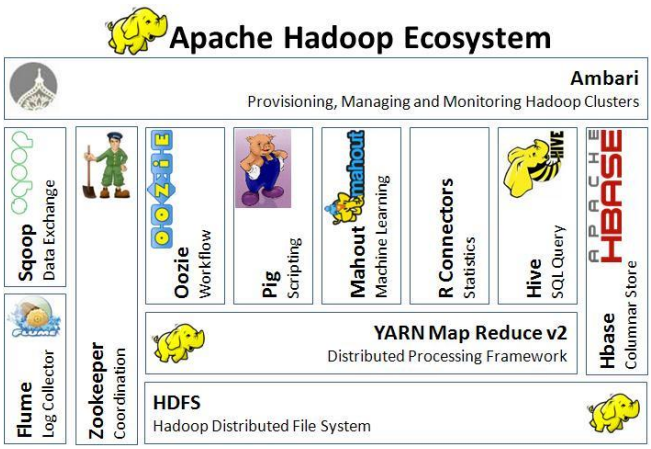


Fig. 2. Hadoop Ecosystem.

Hadoop is a distributed system that uses a Master-Slave architecture. In addition, it provides a distributed file system for storing data called the Hadoop Distributed File System (HDFS) and the Map Reduce programming paradigm to perform the calculations.

Hadoop is a very diverse ecosystem, which grows day after day. Hadoop has grown into a large family of solutions for the storage, management, interaction and analysis of large data, integrated into a rich open

source ecosystem created by the community of the Apache Software Foundation (Fig. 2).

2.2. Job schedulers in Hadoop

The aim of job scheduling included in hadoop [15] is to enable faster processing of jobs and to reduce the response time as much as possible by using better techniques for scheduling depending on the jobs, along with the best utilization of resources. FIFO scheduling is default scheduling mode in Hadoop; the jobs coming first get higher priority than those coming later. In some situations, this type of scheduling has a disadvantage, that is, when longer jobs are scheduled prior to shorter jobs, it leads to starvation.

Fair scheduling shares the resources equally among all jobs. Capacity scheduling was introduced by Yahoo. It maximizes the utilization of resources and throughput in Clusters. LATE [16] scheduling policy was developed to optimize the performance of jobs and to minimize the job response time by detecting slow running processes in a Cluster and launching equivalence processes as the background. Facebook uses Delay scheduling to achieve better performance and lower response time for map tasks by applying changes to MapReduce. In deadline scheduler, the deadline constraints are specified by the user before scheduling the jobs in order to increase system utilization.

Resource aware scheduling improves resource utilization; it uses node, master node, and worked node to complete job scheduling. In matchmaking scheduling [17], each node is marked by the locality marker, which ensures that every node gets an equitable chance to seize a local task. Through this scheduling, high data locality and better cluster utilization is achieved.

There have already been a few review papers on job scheduling algorithms for Big data processing. [14] presented a comparative study on job scheduling methods and discussed their strengths and weakness. [18] presented a review on scheduling algorithms and provided guidelines for the improvement of scheduling algorithms in Hadoop MapReduce. [19] presented a survey on Big data management and discussed various scheduling algorithms in Hadoop. They also discussed the latest advancements related to scheduling algorithms. [20] presented a paper on the scheduling policies for Hadoop and performed a comparative study on MapReduce optimization techniques.

2.3 Major challenges for job scheduling in Big data

There is a clear need for efficient scheduling algorithms for the management of Big data on various nodes in Hadoop clusters, in particular for supporting the execution of machine learning and data mining algorithms. There are various factors that affect the performance of scheduling policies such as data volume (storage), format of data sources (data variety), speed (data velocity), security and privacy, cost, connectivity, and data sharing. To achieve better utilization of resources and management of Big data, new scheduling policies should be designed. The next sections describe some of the main challenges that face job scheduling.

2.3.1. Data volume (storage)

An important problem of Big data is that it is very huge and includes data that is in unstructured formats, which makes it difficult to organize the data for analysis. Data includes both structured and unstructured data; the storage of unstructured data is not an easy task.

2.3.2. Format of data sources (data variety)

Data comes into Big data from various homogeneous as well as heterogeneous resources, and this causes many problems due to the heterogeneity of data resources, data format, and infrastructure

2.3.3. Speed (data velocity)

Today, everybody expects everything to be done instantaneously. The speed is an important issue in Big data. Speed of Big data is restricted by various problems such as query/retrieval problem, import/ export problem, real time/offline problem, and statistical analysis problem.

2.3.4. Connectivity and data sharing

Data sharing and connectivity are still issues that need to be considered. At present, a majority of the data points are not yet connected. There are various issues in Big data related to connectivity and data sharing such as data standard and interfaces, access permissions, and shared protocols.

2.3.5. Cost

Cost is also an issue in Big data. The cost up-gradation issues in Big data are cost comparison between master and slave nodes, and upgrade or modification of nodes.

3. Custom Hadoop Schedulers

Some works have studied the behavior of the schedulers included with Hadoop and proposed improvements [14,21] proposed an analytical model for FIFO and Fair schedulers based on experimental measurements and source code analysis. For a class of Map jobs with heavy-tailed service times, the authors found problems of starvation with the Fair scheduler due to the way of launching Reduce tasks. To reduce that, the Coupling scheduler was proposed that couples the advance of Mappers and Reducers, as well as optimizing the locations for both, achieving mitigation of the problem of starvation and improving the data locality.

In [18] proposed an improvement for the FIFO scheduler that takes into account the location of the data and characteristics of the works. In essence, it adopts different policies for jobs linked to CPU or input / output based on data locality. As a result, it is possible to reduce the data transfer and the execution time of the works. Similarly, [19] designed a scheduler based on dynamic priorities whose objective is to reduce the latency for jobs of variable length. The work focuses on intensive data work, so it tries to maintain the data location during execution.

Other authors have investigated how to deal with Clusters formed by heterogeneous hardware. [20] presented a scheduler that uses the heterogeneity and mix of workloads to optimize jobs that use the same data sets. Although the scheduler has only been simulated, it is based on the idea of looking for the best node of the Cluster to run, based on the idea that a large percentage of MapReduce jobs have the same characteristics in terms of CPU, network and disk usage. The scheduler classifies Cluster jobs as linked to CPU or input / output, similarly the nodes classifies them as good for CPU or input / output and then performs the assignment. [22] proposed a self-adaptive scheduler that takes into account that different nodes require different time to complete the same tasks due to their differences such as processing, communication, architectures and memory. The scheduler uses historical information about each cluster node to adjust execution parameters and discover slow tasks. Thus, you can launch backup tasks in other nodes, also taking into account the data locality. [23] develops criteria to schedule schedulers based on restrictions of deadlines specified by the user and discusses the implementation and preliminary evaluation of a scheduler of Deadlines Restrictions for Hadoop that ensures that only jobs whose deadlines can be met are planned for execution.

In [24] on the contrary, the authors focus on the heterogeneity of the tasks, proposing a scheduler that uses information such as job income rates and average execution times to make better decisions.

In [25] a quantitative approach is adopted where a first detailed study of the behavior of several applications on Hadoop that run on four different hardware configurations, to ensure that the data associated with the jobs are available locally to a cluster in a multi-cluster implementation. On the other hand, the work of [26], observe the changes in the load of the nodes to assign jobs more intelligently.

In [27] the conflict between locality and equity is addressed, and a simple algorithm called delay programming is proposed: when the work that must be scheduled according to equity can not start a local task, it waits a small amount of time, allowing others jobs start tasks in their place. It was verified that the schedule of delays reaches an almost optimal data location in a variety of workloads and can increase performance, while preserving fairness. In addition, the simplicity of delay programming makes it applicable under a wide variety of programming policies in addition to fair exchange.

Finally, [28] studied the interactions between jobs and their intermediate results to group multiple jobs into one and thus reduce the number of queries to access shared intermediate data.

4. Current and Future Research

The purpose of this research work is to study the improvement of time in the exploration and analysis of data sets using an improvement over the schedulers implemented in Hadoop for machine learning and data mining jobs. New schedulers algorithms will also be implemented that will make a better assignment of jobs to the cluster, obtaining advantages in terms of better use of resources and execution times with respect to schedulers such as FIFO schedulers. Machine learning and data mining algorithms have a distinctive characteristic: they involve executing the same algorithm many times over different parts of a input dataset.

The CPU load, network usage and input / output will be analyzed, by executing jobs generated by machine learning and data mining algorithms when executed on Hadoop with large and publicly available datasets. It is intended in a first stage to derive models or profiles of resource use of the aforementioned works considering a space of variability in the data sets and using two Clusters with different hardware characteristics.

The methodology will be similar to that of [29, 30,31] where once the characteristics of the works have been outlined, a simulator developed by the working group will be adapted in order to analyze planning alternatives that improve the execution times and resource utilization. Similar to [32], where

the scheduler uses estimated information on availability of resources to achieve better performance in the execution of tasks in desktop Grids, in this plan estimates will be used on the tasks, starting from the base that the algorithms have well-known behavior and temporal/spatial complexity.

The starting point to obtain job data and then outline its characteristics will be existing implementations of machine learning and data mining algorithms. A good alternative for this is Mahout[2] that offers Hadoop implementations of Naive Bayes classification, k-means clustering, collaborative filtering and recommendation algorithms based on ALS (Alternating Least Squares), among others. Executions of these algorithms using data sets such as those available in the UC Irvine Machine Learning Repository[3] will provide data on the use of resources and performance. From these data, jobs will be grouped according to their demands, resulting performance, input data characteristics, etc.

In a first stage, optimized schedulers for the profiled jobs will be simulated. Once obtained results in this stage, the improvements in executions in the Cluster available in the UNSJ of San Juan and in the ISISTAN of Tandil will be verified. Simulations / tests will be carried out again until satisfactory solutions are achieved.

5. References

[1] Paul C. Zikopoulos, Chris Eaton, Drik deRoos, Thomas Deutsch, and George Lapis. Understanding Big Data - Analytics for Enterprise Class Hadoop and Streaming Data. 2012.

[2] Alejandro Corbellini, Cristian Mateos, Alejandro Zunino, Daniela Godoy, and Silvia N. Schiaffino.Persisting big-data: The nosql landscape. Inf. Syst., 63:1–23, 2017b. doi: 10.1016/j.is.2016.07.009. URL https://doi.org/10.1016/j.is.2016.07.009.

[3] Daniel S. Katz and Xiaobo Zhou. Leading-edge research in cluster, cloud, and grid computing: Best papers from the ieee/acm {CCGrid} 2015 conference. Future Generation Computer Systems, 72:78 – 80, 2017. ISSN 0167-739X. doi:https://doi.org/10.1016/j.future.2016.09.016. URL http://www.sciencedirect.com/science/article/pii/S0167739X16303557.

[4] A. Tommasel, A. Corbellini, D. Godoy, and S. Schiaffino. Personality-aware followee recommendation algorithms: An empirical analysis. Engineering Applications of Artificial Intelligence, 51: 24–36, 2016.

[5] A. Corbellini, C. Mateos, D. Godoy, A. Zunino, and S. Schiaffino. An architecture and platform for developing distributed recommendation algorithms on large-scale social networks. Journal of Information Science, 41(5):686–704, 2015.

[6] Cristian Mateos, Alejandro Zunino, and Marcelo Campo. Jgrim: An approach for easy gridification of applications. Future Generation Computer Systems, 24(2):99 – 118, 2008a. ISSN 0167-739X. doi: http://dx.doi.org/10.1016/j.future.2007.04.011. URL: http://www.sciencedirect.com/science/article/pii/S0167739X07000854.

[7] Pieter Hijma, Rob V. van Nieuwpoort, Ceriel J.H. Jacobs, and Henri E. Bal. Generating synchronization statements in divide-and-conquer programs. Parallel Computing, 38(1):75 – 89, 2012. ISSN 0167-8191. doi: http://dx.doi.org/10.1016/j.parco.2011.10.007. URL: http://www.sciencedirect.com/science/article/pii/S0167819111001384. Extensions for Next-Generation Parallel Programming Models

[8] Alejandro Corbellini, Daniela Godoy, Cristian Mateos, Silvia Schiaffino, and Alejandro Zunino. DPM: A novel distributed large-scale social graph processing framework for link prediction algorithms. Future Generation Computer Systems, 2017a. En prensa.

[9] M. Arroqui, J. Rodriguez Alvarez, H. Vazquez, C. Machado, Cristian Mateos, and Alejandro Zunino. JASAG: A gridification tool for agricultural simulation applications. Concurrency and Computation: Practice and Experience, 27(17):4716–4740, 2015.

[10] Rob V. van Nieuwpoort, Jason Maassen, Rutger Hofman, Thilo Kielmann, and Henri E. Bal. Ibis: An efficient java-based grid programming environment. In Proceedings of the 2002 Joint ACMISCOPE Conference on Java Grande, JGI '02, pages 18–27, New York, NY, USA, 2002. ACM. ISBN 1-58113-599-8. doi: 10.1145/583810.583813. URL http://doi.acm.org/10.1145/583810.583813.

[11] Craig Chambers, Ashish Raniwala, Frances Perry, Stephen Adams, Robert R. Henry, Robert Bradshaw, and Nathan Weizenbaum. Flumejava: Easy, efficient data-parallel pipelines. SIGPLAN

[2] http://mahout.apache.org

[3] http://archive.ics.uci.edu/ml/index.php

Not., 45(6):363–375, June 2010. ISSN 0362-1340. doi: 10.1145/1809028.1806638. URL http://doi.acm.org/10.1145/1809028.1806638.

[12] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, SOSP '03, pages 29–43, New York, NY, USA, 2003. ACM. ISBN 1-58113-757-5. doi: 10.1145/945445.945450. URL http://doi.acm.org/10.1145/945445.945450.

[13] Ivanilton Polato, Reginaldo RÃc , Alfredo Goldman, and Fabio Kon. A comprehensive view of hadoop research–a systematic literature review. Journal of Network and Computer Applications, 46:1 – 25, 2014. ISSN 1084-8045. doi: http://dx.doi.org/10.1016/j.jnca.2014.07.022. URL: http://www.sciencedirect.com/science/article/pii/S1084804514001635

[14] D. Yoo and K. M. Sim. A comparative review of job scheduling for mapreduce. In 2011 IEEE International Conference on Cloud Computing and Intelligence Systems, pages 353–358, Sept 2011. doi: 10.1109/CCIS.2011.6045089.

[15] J.V. Gautam, H.B. Prajapati, V.K. Dabhi, S. Chaudhary, A survey on job scheduling algorithms in big data processing, in: IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT'15), Coimbatore, 2015, pp. 1–11.

[16] M. Zaharia, A. Konwinski, A.D. Joseph, R. Katz, I. Stoica, "Improving MapReduce performance in heterogeneous 46 environments", in: Proc. 8th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2008, San Diego,USA, Dec. 2008.

[17] He, C., Lu, Y., & Swanson, D. (2011). Matchmaking: A New MapReduce Scheduling Technique. In 2011 IEEE Third International Conference on Cloud Computing Technology and Science (pp. 40–47). IEEE. https://doi.org/10.1109/CloudCom.2011.16

[18] Y. Tao, Q. Zhang, L. Shi, and P. Chen. Job scheduling optimization for multi-user mapreduce clusters. In 2011 Fourth International Symposium on Parallel Architectures, Algorithms and Programming, pages 213–217, Dec 2011. doi: 10.1109/PAAP.2011.33.

[19] P. Nguyen, T. Simon, M. Halem, D. Chapman, and Q. Le. A hybrid scheduling algorithm for data intensive workloads in a mapreduce environment. In 2012 IEEE Fifth International Conference on Utility and Cloud Computing, pages 161–167, Nov 2012. doi: 10.1109/UCC.2012.32.

[20] Sreedhar, C., Kasiviswanath, N., & Chenna Reddy, P. (2018). Performance Enhancement of Hadoop for Big Data Using Multilevel Queue Migration (MQM) Technique (pp. 331–342). https://doi.org/10.1007/978-981-10-8237-5_32

[21] Jyoti V. Gautam, Harshad kumar B. Prajapati, Vipul K. Dabhi, Sanjay Chaudhary, A survey on job scheduling algorithms in big data processing, IEE Conf. Pap. (March 2015), http://dx.doi.org/10.1109/ICECCT.2015.7226035

[22] Jian Tan, Xiaoqiao Meng, and Li Zhang. Delay tails in mapreduce scheduling. In Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '12, pages 5–16, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1097-0. doi: 10.1145/2254756.2254761. URL http://doi.acm.org/ 10.1145/2254756.2254761.

[23] Q. Chen, D. Zhang, M. Guo, Q. Deng, and S. Guo. Samr: A self-adaptive mapreduce scheduling algorithm in heterogeneous environment. In 2010 10th IEEE International Conference on Computer and Information Technology, pages 2736–2743, June 2010. doi: 10.1109/CIT.2010.458.

[24] Kc, K., & Anyanwu, K. (2010). Scheduling Hadoop Jobs to Meet Deadlines. In 2010 IEEE Second International Conference on Cloud Computing Technology and Science (pp. 388–392). IEEE. https://doi.org/10.1109/CloudCom.2010.97

[25] Aysan Rasooli and Douglas G. Down. An adaptive scheduling algorithm for dynamic heterogeneous hadoop systems. In Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research, CASCON '11, pages 30–44, Riverton, NJ, USA, 2011. IBM Corp. URL http://dl.acm.org/citation.cfm?id=2093889.2093893.

[26] Krish, K. R., Anwar, A., & Butt, A. R. (2014). [phi]Sched: A Heterogeneity-Aware Hadoop Workflow Scheduler. In *2014 IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems* (pp. 255–264). IEEE. https://doi.org/10.1109/MASCOTS.2014.40

[27] Hsin-Han You, Chun-Chung Yang, and Jiun-Long Huang. A load-aware scheduler for mapreduce framework in heterogeneous cloud environments. In Proceedings of the 2011 ACM Symposium on Applied Computing, SAC '11, pages 127–132, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0113-8. doi: 10.1145/1982185.1982218. URL http://doi.acm.org/10.1145/1982185.1982218.

[28] Zaharia, M., Borthakur, D., Sen Sarma, J., Elmeleegy, K., Shenker, S., & Stoica, I. (2010). Delay scheduling. In *Proceedings of the 5th European conference on Computer systems - EuroSys '10* (p. 265). New York, New York, USA: ACM Press. https://doi.org/10.1145/1755913.1755940

[29] Tomasz Nykiel, Michalis Potamias, Chaitanya Mishra, George Kollios, and Nick Koudas. Mrshare: Sharing across multiple queries in mapreduce. Proc. VLDB Endow., 3(1-2):494–505, September 2010. ISSN 2150-8097. doi: 10.14778/1920841.1920906. URL http://dx.doi.org/10.14778/1920841.1920906.

[30] Juan Manuel Rodriguez, Cristian Mateos, and Alejandro Zunino. Energy-efficient job stealing for cpuintensive processing in mobile devices. Computing, 96(2):87–117, Feb 2014. ISSN 1436-5057.doi:10.1007/s00607-012-0245-5.URL http://dx.doi.org/10.1007/s00607-012-0245-5.

[31] Matías Hirsch, Juan Manuel Rodriguez, Cristian Mateos, and Alejandro Zunino. A two-phase energy-aware scheduling approach for cpu-intensive jobs in mobile grids. J. Grid Comput., 15(1):55–80, 2017. doi: 10.1007/s10723-016-9387-6. URL https://doi.org/10.1007/ s10723-016-9387-6.

[32] Sergio Ariel Salinas, Carlos García Garino, and Alejandro Zunino. PFS: A productivity forecasting system for desktop computers to improve grid applications performance in enterprise desktop grid. Computing and Informatics, 33(4):783–809, 2014.URL http://www.cai.sk/ojs/index.php/cai/article/view/878

Predicción de nubes a corto plazo para una planta solar a partir de datos históricos

Rafael Caballero[1], Luis F. Zarzalejo[2], Álvaro Otero[1], Luis Piñuel[1] and Stefan Wilbert[3]

[1] *University Complutense de Madrid, 28040 Madrid, Spain*
{rafacr, alvama06, lpinuel}@ucm.es
[2] *Renewable Energy Division. Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT), 28040 Madrid, Spain.*
lf.zarzalejo@ciemat.es
[3] *Institute of Solar Research, German Aerospace Center (DLR), 04200 Tabernas, Spain*.
stefan.wilbert@dlr.de

Abstract [1]

Es este trabajo se aborda el problema de la predicción de radiación global sobre superficie horizontal con alta resolución espacial y temporal (5 minutos) a partir de los datos registrados durante un año en la red radiométrica de alta resolución ubicada en la Plataforma Solar de Almería. En particular se muestra un método capaz de predecir el valor de radiación en los siguientes minutos a partir de los valores de los minutos anteriores. El método emplea el tipo de red neuronal recurrente conocido como LSTM, capaz de aprender patrones complejos y predecir el próximo elemento de una serie temporal. Los resultados muestran una mejora apreciable en la precisión del método con respecto a la predicción basada en el último valor conocido.

Keywords: Solar radiation nowcast, LSTM, supervised machine learning.

1. Introducción

Desde hace ya décadas, la generación de electricidad a partir de energías renovables se ha convertido en una necesidad cada vez más acuciante por la problemática medioambiental asociada a residuos y emisiones que los sistemas de producción convencional ocasionan. Esta producción resulta cada vez más competitiva respecto a los métodos de generación tradicionales y, dentro de del conjunto de energías limpias, la energía solar ha adquirido un papel más relevante con el incremento del número de plantas solares en operación. El constante aumento del número de sistemas de producción eléctrica a partir de energía solar y su integración a las redes de distribución energética tradicionales conlleva la necesidad de prever la energía que se va aportar a la red. Por otro lado, los gestores de las plantas solares también requieren de esa previsión para participar en el mercado de la energía y planificar las operaciones de mantenimiento. En definitiva, la capacidad de predecir el recurso solar disponible es crítica para los operadores de plantas solares, ya que esto puede afectar la gestión de la planta y, en consecuencia, la generación de electricidad resultante.
Es este trabajo se aborda el problema de la predicción de radiación global sobre superficie horizontal con alta resolución espacial y temporal (pocos minutos) a partir de los datos recogidos durante un año en la red radiométrica de alta resolución ubicada en la Plataforma Solar de Almería (www.psa.es). El objetivo es encontrar un método que:
a) Disminuya el error de métodos tradicionales, tales como proponer como predictor el último valor conocido o sobre métodos basados en técnicas estadísticas como ARIMA.
b) Dado que el objetivo a medio plazo es integrar el modelo de predicción en una planta solar real, el método propuesto debe ser capaz de definir un modelo y predecir el siguiente valor en un tiempo breve. Aunque se quisieran predecir valores a varios minutos vista, sería deseable que la predicción se realizara en menos de un minuto, ya que en otro caso estamos desaprovechando un resultado nuevo. Buscamos por tanto un modelo de predicción ligero, capaz de mejorar en particular la predicción constante, que, como veremos resulta una muy buena aproximación a muy corto plazo.

Los modelos de predicción de radiación se suelen agrupar en tres tipos. El primer tipo lo forman los *modelos físicos* basados en ecuaciones matemáticas

que describen la física de la atmósfera [1], por ejemplo utilizando los vectores de movimiento de nubes obtenidos a partir de los vectores de movimiento de nubes [2-3]. En una línea diferente, los *modelos estadísticos* [4-5], establecen relaciones entre las observaciones pasadas y las futuras predicciones. La tercera línea dentro de la que se encuentra nuestra propuesta, también emplea los datos históricos pero a través de modelos de aprendizaje automático tales como las redes neuronales [6-8].

La principal diferencia de nuestro trabajo con los ya mencionados es que éstos se centran en predicciones normalmente de partir de 30 minutos [8], nosotros buscamos predicciones a muy corto plazo, de entre 1 y 10 minutos (nowcasting). Es importante notar que cualquier mejora en la predicción, por pequeña que sea, puede tener un impacto económico, al permitir estimar con más precisión la radiación solar recibida y, por tanto, la energía que se va a aportar a la red.

En la siguiente sección comenzamos presentando el conjunto de datos al que hemos aplicado el método y el preprocesado de estos datos. En la sección 3 buscaremos una posible segmentación que permita establecer una distinción entre periodos con y sin nubes. La sección 4 indica los métodos que hemos elegido para comparar con nuestra propuesta, y el tipo de red neuronal elegida. Los resultados de los experimentos se explican en la sección 5. Para terminar, la sección 6 presenta las conclusiones finales.

2. Conjunto de datos inicial

2.1. Estaciones radiométricas

Las 7 estaciones radiométricas utilizadas en este trabajo, forman parte de la red de estaciones ubicadas en la Plataforma Solar de Almería (PSA-CIEMAT). En 2014 dentro del contexto del Proyecto DNICast (predicción de la radiación solar directa con alta resolución espacio-temporal, http://www.dnicast-project.net) se propuso la creación de una red de estaciones que incorporase un total de 19 estaciones radiométricas de alta calidad que cubrieran una superficie de aproximadamente 0.5 km2 (Fig. 1).

Fig. 1 Localización de las estaciones radiométricas en la PSA

En particular, partimos de los datos de radiación global sobre superficie horizontal registrados por 7 de estas estaciones equipadas con piranómetros termoeléctricos Kipp & Zonnen. La localización de cada una de estas estaciones se recoge en la Tabla 1.

Tabla 1: Datos de las estaciones

Nombre	Frec.	Lat.	Long.	Alt.
BSRN	60 s	37.092	-2.363	490.6
ARFRISOL	60 s	37.094	-2.357	499.6
DISS	5 s	37.098	-2.359	504.4
TSA	1 s	37.093	-2.357	499.1
KONTAS	1 s	37.095	-2.355	505.8
CESA1	60 s	37.095	-2.361	503.4
PSA-HP	10 s	37.091	-2.358	500.0

La configuración de la red permite que diariamente se generen los ficheros con los datos de cada estación (las variables registradas son generalmente: irradiancia global horizontal, difusa, directa normal, temperatura, humedad relativa, velocidad y dirección de viento), y éstos son volcados al servidor central vía FTP de forma automatizada.

En este estudio emplearemos los datos de radiación global horizontal correspondientes año 2015 completo. Los datos se han preparado para su

análisis en una secuencia de 4 pasos.

2.2. Ajustes y truncado temporal

Dado que las estaciones tienen distintas frecuencias de registro, tal y como indica la Tabla 1, optamos por considerar solo la mayor entre ellas, que corresponde a una frecuencia de un valor por minuto.

A continuación, procedimos a sincronizar los relojes de cada estación. En algún caso se producían desincronizaciones, que pudieron ser corregidas, excepto en el caso de la estación DISS que tuvo que ser descartada para el resto del estudio.

La siguiente tarea consistió en delimitar una franja horaria de trabajo común para todos los meses, minimizando así el efecto de la altura solar y la variación del número de horas de sol durante el año. Se ha optado por filtrar los datos en el horario entre las 9 y las 16 horas, incluyendo las 9 horas pero excluyendo a partir de las 16h. Esto reduce nuestro conjunto de datos a 7 horas x 60 minutos vectores de 6 valores al día, totalizando 153 300 vectores en el año.

2.3. Datos ausentes

El segundo paso ha consistido en tratar los datos ausentes. Aunque escasos (<0.01%), la ausencia de datos puede generar un mal funcionamiento o incluso impedir la obtención de resultados en algunos métodos de predicción. Las alternativas usuales para corregir esta situación consisten en o bien borrar las filas defectuosas, o bien completarlas a partir del resto de los datos. En nuestro caso hemos optado por la segunda opción, haciendo que el valor perdido sea reemplazado por la media del resto de los sensores en ese periodo.

2.4. Modelo de Cielo Despejado

El tercer paso ha consistido en aplicar un *modelo de cielo despejado* que permita normalizar la variable radiación solar corrigiendo los efectos del movimiento solar aparente. En nuestro caso hemos seguido el método propuesto en [10]. La Figura 2 muestra un día concreto, inestable en las primeras horas y despejado a partir de las 12:30h. En el eje vertical se representa el índice de cielo claro (cociente entre la radiación global y la radiación de cielo despejado) frente a la hora del día. Para facilitar la visualización se han modificado ligeramente las coordenadas Y de cada estación, que de otra manera quedarían superpuestas.

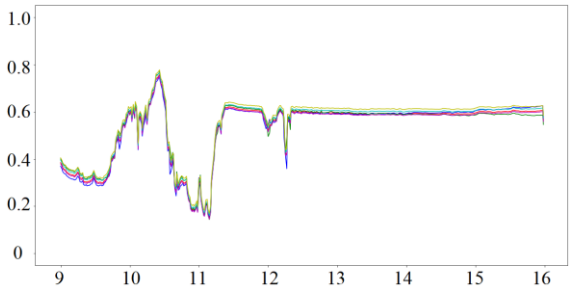


Fig. 2 Gráfica de radiación del día 24 de enero de 2015

2.5. Valores erróneos

Tras aplicar el modelo de cielo despejado, todavía se aplicó una cuarta y última fase de detección de errores ocasionales en las estaciones y su corrección, de nuevo sustituyendo el valor defectuoso por la media de los otros valores.

3. Búsqueda de cielo nuboso mediante segmentación

Un primer análisis de interés consiste en comprobar si los datos admiten alguna segmentación, que permita agrupar mediciones similares y de esta forma determinar un número pequeño de posibles escenarios a analizar.

3.1. Número idóneo de clústers

Uno de los métodos más comunes para este propósito es el método propuesto por T. Caliński y J. Harabasz [11]. En este método, los valores a segmentar se consideran puntos dentro de un espacio n-dimensional euclídeo, para los que hay que encontrar una partición óptima. En nuestro caso tendríamos puntos de un espacio de 6 dimensiones, donde cada dimensión corresponderá al valor de una de las estaciones.

La definición de partición óptima en este método se corresponde a aquella partición que permitan minimizar la suma de cuadrados dentro de cada segmento

Los detalles de este algoritmo están más allá del ámbito de este artículo. En todo caso se encuentra implementado en forma de librerías de los lenguajes Python (librería *sklearn*), y R (librería *vegan*). En ambos casos nos da como número óptimo el valor k=2. Por ejemplo, en R si suponemos que los valores de las estaciones se encuentran en el *dataframe* d:

```
fit <- cascadeKM(scale(d, center = TRUE,  scale =
TRUE), 1, 10, iter = 1000)
```

```
calinski.best <-
as.numeric(which.max(fit$results[2,]))
cat("K óptimo:", calinski.best, "\n")
```

3.2. Proceso de segmentación, e interpretación de los resultados

Los centros de cada segmento no se obtienen directamente con este método, pero una vez obtenido que el K óptimo es 2, podemos emplear algún método de clasificación como k-means. En R podemos escribir simplemente kmeans(d,2), y obtenemos los resultados de la Tabla 2.

Tabla 2: centro de los clúster para cada estación

	Cluster 1	Cluster 2
ARFRISOL	0.259	0.682
CESA	0.264	0.679
PSA	0.269	0.685
TSA	0.268	0.690
KONTAS	0.277	0.703
BSRN	0.286	0.703

La primera columna indica el nombre de cada una de las estaciones, mientras que las otras dos indican el valor central para cada uno de los dos clústers.

La interpretación más natural es que el primer grupo corresponde a una medición de nubes, mientras que el segundo se correspondería a un estándar de cielo despejado. Un vistazo al histograma de frecuencias para cada una de las estaciones parece confirmar esta interpretación. En particular, la Figura 3 contiene el histograma de la estación ARFRISOL.

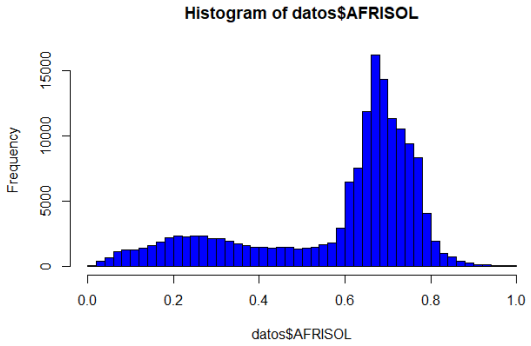


Fig. 3 Histograma de frecuencias para las distintas radiaciones registradas en la estación ARFRISOL

En efecto, la figura parece mostrar una combinación de dos distribuciones. Los centros del clúster 1 se corresponden con el centro de la distribución situada a la izquierda, y por tanto de menor radiación. Esto indicaría que se trata de los valores que corresponden a un cielo con nubes. En cambio, la segunda distribución, más apuntada, corresponde al cielo despejado, que como se aprecia en el histograma es más frecuente en las estaciones estudiadas.

4. Predicción de radiación solar global

El método de segmentación descrito en la sección anterior puede emplearse para detectar de forma automática la presencia de nubes, y constituye nuestra primera aportación. En el resto del trabajo presentamos nuestra propuesta para la predicción de radiación solar global horizontal a corto plazo.

4.1. Estacionalidad

Antes de comenzar el análisis conviene comprobar si nuestro conjunto de datos sigue es una serie temporal estacionaria. Se llama de esta manera a aquella serie temporal cuyas propiedades no dependen del momento en el que se observan [12] Este tipo de series son más difíciles de predecir, y si se da el caso, conviene transformar la serie intentando evitar la estacionalidad.

Para comprobar si nos encontramos ante una ser ello hemos utilizado el método aumentado de Dickey–Fuller (ADF) [13]. En nuestro caso, el método que indica que, en efecto nos encontramos ante un conjunto de datos estacionario, tanto considerando el conjunto de datos anual como las series de datos de cada día. Para evitar la estacionalidad, se suele recurrir a la utilización de las diferencias [14], es decir se reemplaza cada valor por el incremento, positivo o negativo, con respecto al dato anterior. Con esta transformación se pierde un dato por día (el primero de la mañana, donde la

diferencia no tiene sentido), pero en nuestro caso esto carece de importancia dado el elevado volumen de datos del que disponemos.

Tras esta transformación se puede comprobar que ahora se obtienen datos estacionarios, de forma más clara en los datos diarios, y por muy escaso margen en el total anual. Esto nos lleva a decidir que emplearemos datos diarios para nuestros modelos.

4.2. **Métodos directos**

Vamos a utilizar dos métodos sencillos de predicción referenciados habitualmente en la bibliografía.

- Predicción naïve: La predicción es simplemente el último valor conocido.
- Media: Se propone como predicción la media del último intervalo conocido.

A menudo, sobre todo en datos con alta estacionalidad, estos métodos sencillos son a menudo los más efectivos a corto plazo. En nuestro caso, donde consideramos horizontes de muy pocos minutos, veremos que el método naïve resulta muy difícil de superar, tal y como veremos en la sección de experimentos.

4.3. **ARIMA**

Los modelos Auto Regressive Integrated Moving Average (ARIMA) [13], utilizan una combinación de p valores antiguos de la variable para predecir el siguiente valor a calcular, combinados con los q errores observados en las anteriores predicciones. A estos dos parámetros p y q se les añade un tercer parámetro d, que indica el número de diferencias de valores que deben realizarse. Este número de diferencias es el que permite convertir el modelo en no estacional, y que mencionamos en la sección 4.1. Para el cálculo de los valores (p,d,q) hemos utilizado la biblioteca de R forecast, y en concreto el método auto.arima, obteniendo unos valores de p=5, d=1 y q=0.

4.4. **Redes neuronales LSTM**

Una red neuronal artificial (ANN por sus siglas en inglés) [15], es un mecanismo de cómputo compuesto por *neuronas artificiales*. Cada neurona recibe una señal de entrada, que procesa y cuya salida reenvía a otras neuronas conectadas a ella.

El número de neuronas de una red, el grafo de conexiones y la función de procesado de cada neurona se deciden durante el diseño de la red. Sin embargo, tanto las funciones como las conexiones entre las neuronas tienen pesos, que se ajustan durante el proceso de aprendizaje.

La red aprende con respecto a un conjunto de entrenamiento, en nuestro caso los datos recogidos de las estaciones durante el día en curso, ajustando los pesos para minimizar una determinada función de coste o de pérdida que relaciona la salida de la red con la salida esperada, en nuestro caso la predicción para un cierto horizonte futuro f.

En nuestro caso, hemos programado la red mediante la librería Python Keras (https://keras.io/) funcionando sobre la librería TensorFlow (https://github.com/tensorflow/tensorflow). Keras admite numerosas funciones de coste (parámetro *loss*), entre las que hemos elegido la función '*mean_squared_error*', dado que emplearemos la raíz del error cuadrático medio (RMSE por sus siglas en inglés) para determinar el error de cada método.

En las redes neuronales, las neuronas se agrupan en capas. Cada capa realiza algún tipo de transformación sobre su entrada, y a su vez pasa su salida a la siguiente capa. Por tanto, las señales viajan desde la capa de entrada hasta la capa de salida, posiblemente tras transitar por cierto número de capas ocultas.

En nuestro caso, partimos de una capa de entrada que recibe una única señal, y tenemos una única capa oculta, una capa de tipo LSTM (Long Short-Term Memory networks [16], el tipo de red neuronal recurrente (esto es, con ciclos) utilizada a menudo para la predicción de series temporales [17].

Hemos encontrado que el número de neuronas adecuado depende del horizonte considerado. En particular, en los experimentos, hemos empleado una red LSTM de 4 neuronas.

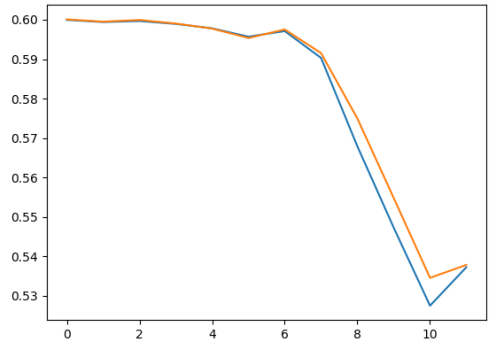


Fig. 4 Predicción por LSTM para los 11 minutos anteriores a las 12h del día 24 de enero de 2015

La Figura 4 muestra un ejemplo de predicción mediante LSTM (línea naranja) comparado con los valores reales (línea azul) para un horizonte de 11 minutos, a partir de los datos recogidos entre las 9 y las 11:49 del día 24 de enero de 2015.

5. Resultados experimentales

La Tabla 3 muestra el resultado de nuestros experimentos.

Tabla 3: RMSE de la predicción LSTM modificada, junto con el % de incremento del RMSE del resto de métodos con respecto a LSTM modificado

f (min)	LSTM (RMSE)	AVG (% inc)	Naïve (% inc)	ARIMA (% inc)
1	0.054	6.59	7.08	7.10
2	0.070	7.46	2.03	1.17
3	0.093	2.31	1.14	1.27
4	0.11	3.72	1.05	1.15
5	0.12	3.02	1.03	1.16
10	0.15	2.47	0.96	1.17
20	0.21	1.37	0.93	1.02

La primera columna indica distintos horizontes de predicción en minutos. La segunda columna (RMSE) muestra la raíz del error cuadrático medio para la red neuronal basada en LSTM. El resto de las columnas muestra el % de diferencia entre el RMSE para el resto de los métodos y el obtenido para la red LSTM. En particular, la tercera columna muestra el incremento de error muestra que el método de la media (AVG) presenta un error de entre el 7.46% (predicción a 2 minutos) y 2.47% (predicción a 10 minutos) el error con LSTM. La cuarta columna corresponde al incremento de error para el método Naïve. Este método, el más sencillo, es el de menor error a partir del minuto 10. De todas formas, en los primeros minutos se ve superado por nuestra propuesta, la red LSTM, aunque por valores de tan solo el 3% ya en el minuto 5. Por otra parte, la predicción basada en ARIMA se sitúa en alrededor del 15% de incremento de error.

Los datos se han obtenido como la media de 3 experimentos por cada día:

- Predicción de radiación en el minuto 12h+f a partir de los datos obtenidos entre las 9 a 12 (f en minutos y con los valores indicados en la tabla 3).
- Predicción de radiación a las 13h+f a partir de los datos obtenidos entre las 12 y las 13 horas.
- Predicción de la radiación a las 15h+f a partir de los datos obtenidos entre 13 y las 15 horas.

Hemos comprobado que los datos no varían sensiblemente al considerar como conjunto de entrenamiento valores superiores a la hora. Es decir, no hay variación significativa entre considerar predicciones a partir de datos acumulados de la hora anterior y considerando, por ejemplo, las 3 horas anteriores. Esto es positivo porque indica que se pueden comenzar a predecir valores a partir de las 10h de cada día. En cambio, con datos de menos de una hora sí se aprecian aumentos significativos del error.

En resumen, la tabla muestra la media de un total de 365 (días) x 3 (predicciones por día) x 6 (estaciones) = 6570 tests.

Para comprobar si los datos de la tabla suponen diferencias significativas hemos llevado a cabo la prueba de los rangos con signo de Wilcoxon [18], comprobando que, en efecto, las diferencias de la media de RMSE entre LSTM y cada una de las

otras técnica muestra diferencias estadísticamente significativas.

Por último, y en relación con el objetivo de lograr un método predictivo eficiente en términos de tiempo, debemos señalar que los modelos se generan en media en un tiempo de 23 segundos en un ordenador XPS 13 9350, con 4 CPUs a 2.20 gigahercios y 16 gigabytes de RAM, por tanto cumpliendo nuestro objetivo de ser capaz de general los modelo en un tiempo menor al minuto.

6. Conclusiones

En este artículo hemos considerado el problema de la predicción de radiación directa a muy corto plazo sobre un conjunto de datos concreto recopilados en la Plataforma Solar de Almería durante un periodo de un año. Hemos comprobado que, mediante la utilización de redes LSTM, es posible mejorar, aunque por un margen estrecho, la predicción basada en la repetición del último valor conocido, método simple pero muy efectivo en los primeros minutos, y que sí mejora los resultados de otras técnicas habituales como la predicción basada en la media, o incluso de otras más complejas como ARIMA.

En cuanto a trabajo futuro, debemos señalar que nuestra red solo considera cada estación individualmente, es decir la predicción está basada en los datos de la propia estación durante la hora anterior. Pensamos que el desarrollo de modelos multidimensionales que empleen los resultados de todas las estaciones para predecir los de una en concreto pueden llevarnos a predicciones más precisas. La intuición es que la radiación en un punto puede venir anticipada por la de estaciones cercanas sobre las que, por ejemplo, ya ha comenzado a pasar una nube.

7. Acknowledgements

Este trabajo ha sido subvencionado parcialmente por el proyecto nacional español TIN2015-66471, por el proyecto Santander-UCM PR26/16-21B-1.

The authors declare that no competing interests exist.

8. References

[1] D. Renne, *Semi-Annual Status. Task 36: Solar. Resource Knowledge Management*. Solar Resource Knowledge Management. 2009.

[2] E Lorenz, A Hammer, and D Heinemann. *Short term forecasting of solar radiation based on satellite data*. In EUROSUN 2004 (ISES Europe Solar Congress), pages 841- 848, 2004.

[3] Bosch, J. L. y Kleissl, J. Cloud motion vectors from a network of ground sensors in a solar power plant. Solar Energy 95(1), 13-20, 2013.

[4] G. Reikard. *Predicting solar radiation at high resolutions: A comparison of time series forecasts*. Solar Energy, 83(3):342-349, 2009.

[5] Martín, L., Zarzalejo, L. F., Polo, J., Navarro, A., Marchante, R. y Cony, M. Prediction of global solar irradiance based on time series analysis: Application to solar thermal power plants energy production planning. Solar Energy 84(10), 1772-1781, 2010.

[6] C. Paoli, C, Voyant, M. Muselli, and M.L. Nivet. *Forecasting of preprocessed daily solar radiation time series using neural networks*. Solar Energy, 84(12), 2146-2160, 2010

[7] A. Mellit and A. M. Pavan. *A 24-h forecast of solar irradiance using artificial neural network: Application for performance prediction of a grid-connected PV plant at Trieste, Italy*. Solar Energy, 84(5), 807-821, 2010.

[8] M. Bou-Rabee, S. A. Sulaiman, M. Saleh, and S. Marafi. *Using artificial neural networks to estimate solar radiation in Kuwait*. Renewable and Sustainable Energy Reviews, 72, 434-438. 2017.

[9] M. Diagne, M. David, P. Lauret, J. Boland, and N. Schmutz. *Review of solar irradiance forecasting methods and a proposition for small-scale insular grids.* Renewable and Sustainable Energy Reviews, 27, 65-76. 2013.

[10] I. A. Walter, R. G. Allen, R. Elliott, M.E. Jensen, D. Itenfisu, B Mecham, ... and T Spofford. *ASCE's standardized reference evapotranspiration equation.* In Watershed Management and Operations Management 2000 (pp. 1-11). 2000.

[11] T. Caliński, and J. Harabasz *A dendrite method for cluster analysis*. Communications in Statistics-theory and Methods, 3(1), 1-27. 1974

[12] D. Kwiatkowski, P. C B Phillips, P. Schmidt, and Y. Shin. 1992. *Testing the Null Hypothesis of Stationarity Against the Alternative of a Unit Root: How Sure Are We That Economic Time Series Have a Unit Root?* Journal of Econometrics 54 (1-3): 159–78. 1992.

[13] R. J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts. 2018.

[14] S. Makridakis, S.C. Wheelwright, and R.J. *Hyndman. Forecasting: methods and applications*. John Wiley & Sons, 1998.

[15] S. Haykin. *Neural Networks: A Comprehensive Foundation Upper*. Saddle River. NJ, USA, pp. 1–842. 1998.

[16] S. Hochreiter and J. Schmidhuber. *Long Short-Term Memory*. Neural Computation. 9 (8): 1735–1780. 1997

[17] F. A. Gers , J. Schmidhuber and F. Cummins. *Learning to Forget: Continual Prediction with LSTM*. Neural Computation Volume 12-10. p.2451-2471. 2000

[18] F. Wilcoxon. *Individual Comparisons by Ranking Methods*. Biometrics 1, 80-83. 1945

Treatment of Massive Metagenomic Data with Graphs

Cristóbal R. Santa María[1], Romina A. Rebrij[2], Victoria Santa María[3] and Marcelo A. Soria[4]

[1] *Departamento de Ingeniería-Universidad Nacional de La Matanza, San Justo, Buenos Aires, B1754JEC, Argentina*
csantamaria@unlam.edu.ar

[2] *Ph.D. Program, Facultad de Ingeniería, Universidad de Buenos Aires, Buenos Aires, C1063ACV, Argentina*
rominarebrij@gmail.com

[3] *Instituto de Investigaciones Médicas "Alfredo Lanari" - Universidad de Buenos Aires, Buenos Aires, C1427ARO, Argentina*
vctrsntmr@gmail.com

[4] *Microbiología – Facultad de Agronomía - Universidad de Buenos Aires, Buenos Aires, C1417DSE, Argentina*
soria@agro.uba.ar

Abstract

Among the *de novo* strategies to assemble metagenomic DNA fragments the application of de Bruijn graphs stands out. These graphs greatly reduce the computational complexity and overload that arises as a consequence of the huge data volume. An Eulerian cycle can be established on a de Bruijn graph that allows the assembly of sequence reads into longer fragments for genome reconstruction. This paper shows the theoretical principles of the computational schema applied. Also, the difficulties that appear in the practical application of the method and the algorithmic features of some of the available open source programs. Finally, the work of the authors research group is summarized.

Keywords: high-throuput DNA, sequencies assembly, metagenomics, graphs, eulerian cycle

1. Introduction

Metagenomics is a branch of biology that studies deoxyribonucleic acid (DNA) extracted from environmental samples, mainly of microbial origin. Upon extraction, the DNA is fragmented and sequenced using some high-throughput sequencing technology. This approach has several advantages, among them two stand out. First, the steps between sampling and data production are minimized, which reduces the number of perturbations introduced by laboratory manipulations. Second, metagenomic analyses do not require the isolation of pure cultures nor their growth in laboratory culture media. This feature allows the detection and quantification of the huge number of microorganism that cannot be cultivated and thus are undetectable through traditional techniques. One of the drawbacks of the sequencing techniques is that they produce fragments, also called reads, that are relatively small compared to a full bacterial genome. The most popular sequencing technology nowadays is Illumina which produces fragments that vary depending the specific protocol between 36 and 250 base pairs of DNA, while a typical bacterial genome range in size between 3 and 6 gigabases. Some newer technologies, such as PacBio or Oxford Nanopore, can reach fragment sizes of several thousand kilobases, which facilitates the process but they are still considerably shorter than the full genome. There are two types of strategies to reconstruct a genome or at least long fragments of it. The first is to map the reads against a similar reference genome. This option is fast and quite accurate but is also possible when a very similar genome is available. The second, known as the *de novo* strategy is to assemble the genome matching and joining reads in progressively longer fragments, or contigs. Although in theory it would be possible to assemble all reads into contigs through a brute force approach consisting of matching all reads vs. all, as the number of reads increases it becomes computationally unfeasibly. Several strategies and heuristics were proposed over the years and many of the successful attempts were based on applications of graph methods, in special De Bruijn graphs [1]. In this work we analyze how the assembly of DNA sequences is achieved using graphs, we also show how these techniques extract information from the pools of reads while reducing the overall requirements of memory.

2. Complexities of contig assembly

Current DNA sequencing technologies can generate 10^6 - 10^{12} reads (fragments) with sizes that typically vary between 50 and 250 base-pairs (bp). In the case of bacterial genomes this short reads must be ordered and assembled into circular DNA molecules with usual lengths ranging between 1 and 6 Mbp, but can be as long as 14 Mbp. Ideally, every single position of the genome would be covered by multiple reads, and every read should have other reads with which it would overlap partially or completely (Fig. 1).

Thus, a brute force approach of finding overlapping

```
---ACCGT--
-----CGTGC
CCTACC----
-CTACCGT-

CCTACCGTGC
```

Fig. 1. Optimal alignment for sequence reconstruction

reads should accomplish the task or sorting and assembling the genome. However, the huge size of the sequencing data prevents the implementation of such an approach. Besides, there are certain genomic features (repetitive sequences, duplicated genes, etc.) that cannot be sequenced properly or that introduce uncertainties in the ordering of reads. Besides, the sequencing methodology introduces errors. In consequence the result produced by an assembler is not a single strand of DNA per chromosome, but a set of contigs built from overlapping reads (Fig. 2).

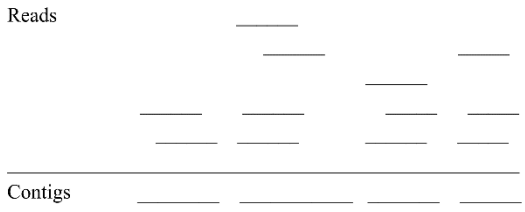


Fig. 2. Optimal alignment for sequence reconstruction

To convert the assembly of genomes into a more tractable problem, several strategies based on the use of graph theory were proposed [2].

One variant is to consider that each sequence read is fragmented into overlapping strings of length *k*, known as *k-mers*. So that two-adjacent *k-mers* overlap by *k-1* nucleotides. Then a graph is built considering that each *k-mer* is a node and the edges are the overlaps of length k-1 between adjacent *k-mers*. For example, consider the toy circular genome in Fig. 3 (left), if the sequence reads generated by the sequencer (Fig. 3, right) are divided into k-mers of length three, the list of all 3-mers is ACG, CGC, GCC, CCG, CGT, GTC, TCG, CGA, GAA, AAC. After that, all two-base overlaps between pairs of list elements could be searched, counted and organized in a Hamiltonian cyclic graph (Fig. 4) in which nodes represent the k-mers, and the directed edges represent the k-1 mers of the overlap.

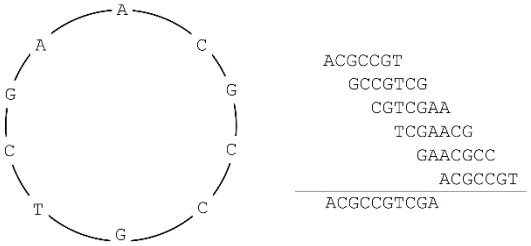


Fig. 3. Circular genome (left) and reads obtained from it and the resulting contig (right)

In other words, the k-mer is formed by removing the prefix from the source node and suffix of the destination node. In this way, by moving from node ACG to node CGC the sequence ACGCC is obtained. By completing this walk along the cyclic graph, the whole sequence of the toy genome (ACGCCGTCGA) is reconstructed.

As a more real example, we could consider the 10^6 – 10^9 million 150-bp single-end reads that a typical Illumina run generate assume that the assembly would be carried out using k-mers of length length 55. Finding the Hamiltonian cycle under these conditions would require between 10^{12} and 10^{18} comparisons of 55-base pairs of k-mers. This is a currently unsolvable NP-complete problem.

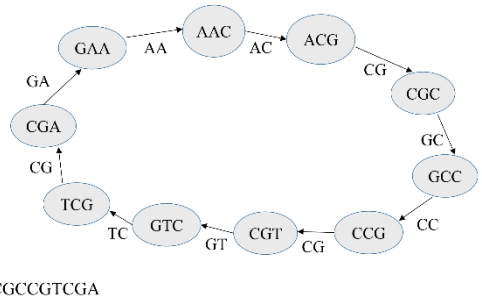


Fig. 4. Hamiltonian cycle and reconstructed genome sequence

In contrast, the application of the De Bruijn graphs renders the problem solvable. Nicolaas De Bruijn extended the results that Leonhard Euler obtained in 1741 while analyzing the famous seven bridges problems [3]. In short, Euler found the conditions for the existence of what was later named the Eulerian cycle. In this cycle Euler considered that the bridges were edges of a graph that connected different nodes representing the different city neighborhoods. His goal was to determine whether there was a path that started and finished in the same node and visited

every other node traversing each edge at most once. Euler determined that:

a) If there are more than two nodes with an odd number of incident edges, the cycle does not exist.
b) If there are only two nodes and they are connected by an odd number of edges, the cycle does exist and can be started from any node.
c) If there is no node with an odd number of incident edges, the cycle exists and can be started from any node.

The graph also must be connected, that is, every node can be visited from any other node in the graph. If the graph is directed, it is required that the number of output edges is equal to the number of input edges for every node. This is known as a balanced directed graph.

In 1946 de Bruijn discovered how to search the shortest circular superstring that contains all substrings of k elements from an alphabet with n elements [1]. Given the four symbols (bases) that comprise DNA, there are $4^3 = 64$ possible 3-mers. If the size of DNA strings is fixed at k=55, as can easily be the case with next generation sequencing technologies, 4^{55} different 55-mers could exist. The application of de Bruijn's idea to DNA sequencing is to consider each directed edge as a k-mer connecting a source node representing the *k-1* prefix of the k-mer, and a target node representing its k-1 suffix [4]. If all k-mers of a DNA sequencing procedure are considered, the resulting Eulerian cycle will represent the shortest superstring that contains every k-mer once. That is, the superstring that contains all the k-1 overlaps. Fig. 5 shows how this graph representation operates on the given example.

The problem of finding an Eulerian cycle was algorithmically solved by Euler itself in 1741 and has, in consequence, a viable computational solution [3]. In the case of sequence assembly, the DNA reads are fragmented into k-mers and connected in a de Bruijn graph. which is then used to find the Eulerian cycles that will constitute the contigs.

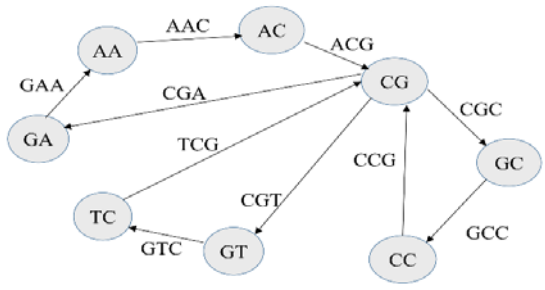


Fig. 5. Eulerian cycle of the de Bruijn graph

3. Application

The practical applications of this assembly method reveal some difficulties. First, the DNA reads may not cover all of the possible k-mers the target genome could produce. Second, sequence duplications occur in most genomes with varying frequencies depending on the organisms under study. This fact determines a multiplicity of the number of inward and outward edges a node can have. Third, the chemical reactions that occur in the sequencing equipment can produce erroneous base calls. These sequencing errors must be addressed before the assembly because they will generate k-mers that do not exist in the genome.

Different software solutions, mainly open-source, have been developed and published in order to solve the difficulties pointed out in the previous paragraph and to include heuristics that accelerate the assembly process. For example, IDBA-UD compare the quality parameters in complementary reads to infer the most likely k-mer [5]. MEGAHIT [6] builds a succinct version of the de Bruijn graph that code m edges using a vector of O(m) bits that marks-up the validity or invalidity of every edge. It starts with suffixes and prefixed of length eight and iteratively increment the graph size by increasing k. This way, in the first iterations edges with errors are removed and later, with larger k's, the sequence repetitiosn are removed. Within the framework of the project "Applications of Data Mining Techniques for the Analysis of the Human Microbiom through Metabolic Functionalities" of the Universidad Nacional de la Matanza, we carried out the assembly of 143 metagenomes obtained from bacterial DNA extracted from stool, rectal swab, and mucosal samples deposited in the NCBI under the Bioproject accession number PRJNA397450. The data from this study was originally analyzed only at the DNA read level, but we are extending it by assembling the metagenomes to determine whether the use of longer contig sequences allows us to detect genes or bacteria that has potential as biomarkers of health/disease conditions.

4. References

[1] N. de Bruijn. "A combinatorial problem," *Proceeding Nederlands Akademiks Wetensch,* vol. 49, pp. 758-764, 1946

[2] D. Coil, G. Jospin and A. Darling, "A5-miseq: an updated pipeline to assemble microbial genomes from Illumina MiSeq data," *Bioinformatics*, Vol. 31, no. 4, pp. 587-589, 2015

[3] L. Euler, "Solutio problematis ad geometriam situs pertinentis," *Commentarii Academiae*

Scientiarum Petropolitanae, vol. 8, pp. 128-140, 1741
[4] P. Campeau, P. Pevzner, and G. Tesler, "How to apply de Bruijn graphs to genome assembly," *Nature Biotechnology,* vol. 29, no. 11, pp. 987-991, 2011
[5] Y.Peng, H. Leung, S. Yiu, and F. Chin, " IDBA-UD: de novo assembler for single-cell and metagenomic sequencing data with hibhly uneven depth," *Bioinformatics,* vol. 28, no. 11, pp. 1420-1428, 2012
[6] D. Li, C. Liu, R. Luo, K. Sadakane, and T. Lam, "MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph," *Bioinformatics*, vol. 31, no.10, pp. 1674-1676, 2015

Análisis Simbólico de Datos:una potente herramienta para Big Data

Adriana Mallea[1],Myriam Herrera[2], and María Inés Lund[2]

[1]*Departamento de Matemática, Universidad Nacional de San Juan, San Juan, Argentina*
[2]*Instituto de Informática, Universidad Nacional de San Juan, San Juan, Argentina*

E-mail: lamallea@ffha.unsj.edu,ar; {mherrera, mlund}@iinfo.unsj.edu.ar

Abstract Los datos simbólicos, introducidos por Edwin Diday en los ochenta, se ocupan del análisis de datos con variabilidad intrínseca que debería ser tenida en cuenta. En minería de datos, análisis multivariado de datos y estadística clásica los elementos analizados generalmente son entidades individuales, para las cuales se graba un valor individual de cada variable. Por ejemplo, individuos descriptos por edad, salario, nivel educativo, etc. Pero cuando los elementos de interés son clases o grupos de algún tipo, como los ciudadanos que viven en una ciudad determinada, modelos de autos en lugar de vehículos específicos, etc.; hay variabilidad inherente en los datos. Reducir esta variabilidad mediante medidas de tendencia central, tales como media aritmética, mediana o moda, lleva obviamente a una pérdida de información importante.

El análisis de datos simbólicos proporciona un marco que permite representar datos con variabilidad, usando nuevos tipos de variables. Los datos simbólicos se pueden representar usando los arreglos usuales en forma de matrices, pero en los cuales los elementos de cada celda no son valores numéricos reales individuales, sino conjuntos finitos de valores, intervalos o, de forma más general, distribuciones.

En los últimos años surgió el término **Big Data**, refiriéndose a conjuntos de datos tan grandes y complejos que se vuelven difíciles de procesar, en un tiempo razonable, con aplicaciones tradicionales de análisis de datos. El análisis simbólico de datos, al ofrecer la posibilidad de agregación de datos al nivel de granularidad elegido por el usuario mientras se mantiene la información sobre la variabilidad intrínseca, desempeña un papel importante en este contexto.

En el presente trabajo se desarrollan algunos conceptos fundamentales de la teoría de objetos simbólicos y se emplean metodologías del análisis de tales objetos en la Encuesta Permanente de Hogares, correspondiente al tercer trimestre del año 2016. En particular se trabaja con los hogares del Gran San Juan, región de Cuyo con dos propósitos: caracterizar a éstos hogares y comparar a los encuestados en cuanto a su estado social, en relación a variables de interés, en particular su Nivel de Estudios. Para el primer objetivo se define como objeto simbólico el hogar; mientras que para el segundo propósito se trabaja con objetos de tipo eventos. Se destaca la posibilidad de trabajar con bases individuales relacionales a fin de responder a diferentes propósitos de estudio, como también la generación de objetos simbólicos que son descriptos por variables simples, multivaluadas, probabilísticas y del tipo intervalo.

Con este trabajo se logra comprobar que los Datos Simbólicos son una herramienta de gran utilidad para el manejo y análisis de grandes volúmenes de datos, por lo que consideramos una herramienta fundamental para dos grandes áreas de la Ciencia de datos: Data Mining y Machine Learning.

Keywords Objeto simbólico, Variables Simbólicas, Data Mining.

1 Introducción

El Análisis de Datos Simbólicos permite la extensión de la Estadística a la Estadística de las intenciones o conceptos y más concretamente la extensión de problemas, métodos y algoritmos de análisis de datos clásicos a datos simbólicos. Según Diday el Análisis de Datos Simbólicos crea un puente entre la Estadística y

el Aprendizaje Automático.

Diday [1,2] formaliza los conceptos de intención y extensión, debidos a Arnauld y Nicole (Arnauld y Nicole, 1662). La intención de un concepto constituye su descripción, mientras que la extensión es el conjunto de individuos cuya descripción es acorde a la del concepto. La intención, que se representa por un objeto simbólico, se describe por los datos simbólicos y por un mecanismo de reconocimiento de los individuos de la extensión.

Diday introduce los objetos simbólicos y presenta una formalización que permite tratar conocimientos más ricos que los datos habituales y establece una relación con el modelo clásico de Análisis de Datos [3]. Un objeto simbólico representa una intención, un concepto y se define, en términos generales, como una conjunción de valores, o conjuntos de valores que pueden ser ponderados. Constituye una descripción en intención de una clase de individuos que constituyen la extensión.

Los objetos simbólicos representan conceptos, entendidos como la intención y extensión del mismo. La intención de un concepto representa las propiedades que lo definen y que lo hacen distinto de los demás conceptos. La extensión de un concepto se compone de los individuos que se definen por el concepto o que cumplen las propiedades que definen el concepto. Se describen por variables y datos simbólicos y proporcionan un mecanismo de vuelta a bases de datos o conjuntos de individuos en el sentido de conocer aquéllos que se adecuan o relacionan con las descripciones simbólicas representadas por los objetos (las intenciones), según determinadas relaciones que también forman parte de las intenciones.

El Análisis de Datos Simbólicos es una generalización de las técnicas de Análisis de Datos aplicadas a matrices de datos simbólicos. Las definiciones y notación de variables simbólicas y objetos simbólicos han estado en constante evolución desde sus inicios [1, 2, 3, 4, 5].

2 Datos Simbólicos

2.1 Preliminares

Sea $\Omega = \{\omega_1, ..., \omega_k\}$ un conjunto de individuos, sea $\mathcal{Y}$ un conjunto o dominio de posibles valores observados y $E = \{e_1, ..., e_n\}$ un conjunto de objetos. Como casos particulares más frecuentes se tiene que E es un subconjunto de Ω o un subconjunto de las clases de Ω, es decir, $E \subseteq \Omega$ o $E \subseteq P(\Omega)$. En el segundo caso, los datos simbólicos correspondientes, describen clases de individuos de Ω.

La descripción de un elemento $e \in E$ por una variable con dominio $\mathcal{Y}$ puede darse por: un elemento del conjunto $\mathcal{Y}$ (variable clásica), un subconjunto de elementos del conjunto $\mathcal{Y}$, un intervalo de valores del conjunto $\mathcal{Y}$ donde se ha definido un orden, un subconjunto de elementos del conjunto $\mathcal{Y}$ donde cada uno de ellos es ponderado por un peso o modo.

La matriz de datos en el Análisis Datos Simbólicos es la matriz $[X]$, cuyas filas representan n unidades u objetos del conjunto E descriptos por un vector de variables simbólicas $X = (X_1, ..., X_p)$. Es decir, las celdas de una fila se corresponden con los datos simbólicos descritos por el vector X aplicado a un elemento de E.

A continuación, se definen los distintos tipos de variables y datos simbólicos.

Variable conjunto valuada

Definition 1. *Se dice que X es una variable conjunto valuada si es una aplicación:*

$$X : E \to P(\mathcal{Y})$$
$$e \to X(e)$$

- $X(e)$ es la descripción de un elemento $e \in E$ en $P(\mathcal{Y})$ dada por la variable X

- $P(\mathcal{Y})$ es el conjunto de descripciones de los elementos de E.

En caso que $|X(e)| = 1$ (donde $|A|$ denota la cardinalidad del conjunto A) para todo $e \in E$ se denomina monovaluada. Se llama multivaluada se $1 < |X(e)| < \infty$ para todo $e \in E$. Se puede tratar de una variable categórica o cuantitativa.

Se puede extender la definición anterior al caso multivariante.

Variables modales probabilistas

Sea $\mathcal{Y} = \{z_1, ..., z_x\}$ y sea $\mathcal{M}(\mathcal{Y}) =$

$\{q : q$es una distribución de probabilidad definida en $\mathcal{Y}\}$,

el conjunto de descripciones modales probabilistas de elementos de E. Una descripción $q \in \mathcal{M}(\mathcal{Y})$ se define como:

$$q : \mathcal{Y} \to [0,1]$$
$$z_i \to q(z_i) \quad \text{con} \sum_{i=1,...,x} q(z_i) = 1$$

Se identifica el **dato simbólico** o descripción simbólica q con

$$q \equiv (z_1 q(z_1), ..., z_x q(z_x))$$

Definition 2. *Se dice que X es una variable modal probabilista definida en E, si es una aplicación*

$$X : E \to \mathcal{M}(\mathcal{Y})$$
$$e \to X(e) = q_e$$

tal que dado $e \in E$ le asocia $X(e) = q_e$ donde q_e es una distribución de probabilidad en el conjunto $\mathcal{Y}$ de posibles valores de observación, completado por una $\sigma - álgebra$

$X(e)$ es la descripción modal probabilista (en $\mathcal{M}(\mathcal{Y})$) del elemento $e \in E$ dada por la variable modal probabilista X.

En el caso en que $E \subseteq \mathcal{P}(\Omega)$, la variable X es un descriptor modal probabilista de clases de individuos de Ω y $\mathcal{M}(\mathcal{Y})$ el conjunto de las descripciones modales probabiliastas de clases de Ω, o de los elementos de $\mathcal{P}(\Omega)$.

La definición de variable modal probabilista se puede extender a una variable modal cuyos modos asociados a las categorías de $\mathcal{Y}$ son frecuencias o pesos. Así también al caso multivariante.

2.2 Objetos Simbólicos

Sea el conjunto $E = \{e_1, ..., e_n\}$ de elementos descriptos por p variables simbólicas $X_1, ..., X_p$ definidas en E con dominios finitos $\mathcal{Y}_1, ..., \mathcal{Y}_p$. En general la variable simbólica X_j es una aplicación $X_j : E \to \mathcal{D}$, siendo $\mathcal{D}$ un conjunto de descripciones de elementos de E asociado al conjunto o dominio $\mathcal{Y}$. El conjunto $\mathcal{D}$ puede ser: $\mathcal{Y}$ en el caso de que X_j sea una variable monoevaluada, $\mathcal{P}(\mathcal{Y})$ si X_j es una variable simbólica multievaluada o $\mathcal{M}(\mathcal{Y})$ si X_j es modal. Se dice que una descripción $d \in \mathcal{D}$ está asociada al conjunto o dominio $\mathcal{Y}$.
Se definen relaciones de dominio entre las descripciones. En particular para un par de descripciones se tiene:

Definition 3. *Sean $\mathcal{D}$ y $\mathcal{D}'$ dos conjuntos de descripciones de clase asociados a un mismo dominio, $\mathcal{D} \times \mathcal{D}'$ su producto cartesiano, una relación de dominio $\mathcal{R}$ definida en $\mathcal{D} \times \mathcal{D}'$ es una aplicación:*

$$\mathcal{R} : \mathcal{D} \times \mathcal{D}' \to \mathcal{L}$$
$$(d, d') \to \mathcal{R}(d, d') := [d\mathcal{R}d']$$

que a cada par de descripciones $(d, d') \in \mathcal{D} \times \mathcal{D}'$ le asocia un valor, denotado por $[d\mathcal{R}d']$ que mide el grado de adecuación o conexión de ambas descripciones. $\mathcal{L}$ es el conjunto de comparación de descripciones. El valor $[d\mathcal{R}d']$ es el nivel de relación entre las descripciones d y d'.

La relación de dominio es booleana si el conjunto $\mathcal{L} = \{0, 1\}$. Si $\mathcal{L} = [0, 1]$, se dice que $\mathcal{R}$ es una relación difusa.

Definition 4. *Sea una colección de relaciones de dominio $(\mathcal{R}_1, ..., \mathcal{R}_p)$, cada $\mathcal{R}_j$ definida en el producto cartesiano $\mathcal{D}_j \times \mathcal{D}'_j$.Sean $\mathcal{D} := \mathcal{D}_1 \times ... \times \mathcal{D}_p$ y $\mathcal{D}' := \mathcal{D}'_1 \times ... \times \mathcal{D}'_p$ los correspondientes productos cartesianos de los conjuntos de descripciones. La relación producto $\mathcal{R} = \mathcal{R}_1 \times ... \times \mathcal{R}_p$ definida en $\mathcal{D} \times \mathcal{D}'$ es la aplicación:*

$$\mathcal{R} : \mathcal{D} \times \mathcal{D}' \to \mathcal{L}$$
$$(d, d') \to \mathcal{R}(d, d') := [d\mathcal{R}d'] :=$$
$$g\left(\left\{\left[d_j \mathcal{R}_j d'_j\right], j = 1, ..., p\right\}\right) = \bigwedge_{j=1,...,p} \left[d_j \mathcal{R}_j d'_j\right]$$

que a cada par de descripciones $(d, d') \in \mathcal{D} \times \mathcal{D}'$, $d = (d_1, ..., d_p)$, $d' = \left(d'_1, ..., d'_p\right)$ le asocia un valor, denotado por $[d\mathcal{R}d']$. La aplicación g es una aplicación simétrica, que en general es el operador conjuntivo lógico estándar.

Definition 5. *Un objeto simbólico de tipo evento en E es una t-upla $(a, \mathcal{R}, d)$ donde:*

- *a es una función, denotada por* $a = [X\mathcal{R}d]$*, con* X *una variable simbólica con dominio* $\mathcal{Y}$ *definida por* $X : E \to \mathcal{D}$*,* $\mathcal{D}$ *un conjunto de descripciones de elementos de* E*, asociado al conjunto* $\mathcal{Y}$*. La función* $a : E \to \mathcal{L}$ *es tal que a cada elemento* $e \in E$ *le asocia el nivel de relación de su descripción en* $\mathcal{D}$ *(dada por* X*) con la descripción* d*.*
- $\mathcal{R}$ *es una relación de dominio definida en* $\mathcal{D} \times \{d\}$
- d *es una descripción de un conjunto de descripciones asociado al conjunto* $\mathcal{Y}$*.*

Definition 6. *Sea* $(a, \mathcal{R}, d)$ *con* $a = [X\mathcal{R}d]$*, un evento booleano definido en* E*. Se llama extensión del evento booleano* a *en* E *y se denota por* $Ext_E(a)$*, al subconjunto de elementos de* E *cuya descripción en* $\mathcal{D}$ *(dada por* X*) se relaciona con el evento* a*:*

$$Ext_E(a) = \{e \in E : a(e) = [X(e)\mathcal{R}d] = 1\}$$

Sea $(a, \mathcal{R}, d)$ *con* $a = [X\mathcal{R}d]$*, un evento definido en* E *y* $\delta \in [0, 1]$*. Se llama extensión de nivel* δ *del evento* a *en* E *y se denota por* $Ext_{E,\delta}(a)$*, al subconjunto de elementos de* E *cuya descripción en* $\mathcal{D}$ *(dada por* X*) tiene un nivel de relación con el evento* a *igual o superior a* δ *:*

$$Ext_{E,\delta}(a) = \{e \in E : a(e) = [X(e)\mathcal{R}d] \geq \delta\}$$

Existen objetos simbólicos tipo aserción, es decir referidos a varias variables. Se compone de varios eventos y está dotada de una función combinación de niveles de relación. Esta función combina los niveles de relación de cada uno de los eventos aplicados a un elemento del conjunto sobre el cual está definida la aserción.

3 Aplicación a la Encuesta Permanente de Hogares

Se aplican metodologías del Análisis Simbólico [6, 7, 8, 9],a datos de la Encuesta Permanente de Hogares (EPH), tercer trimestre de 2016, correspondiente al Gran San Juan. La EPH fue proporcionada por el Instituto Nacional de Estadísticas y Censos (INDEC). La encuesta consta de dos cuestionarios: uno, con datos de la vivienda y características del hogar y otro individual, con datos laborales, de ingresos, de educación y de migración de cada uno de los componentes del hogar.

Antes de comenzar el análisis se eliminaron las variables con información redundante, que tuvieran poca información, aquellas variables que dieran información muy detallada y se identificaron los registros que contenían información inconsistente a fin de eliminarlos.

El objetivo es analizar las ventajas del estudio de una gran base de datos mediante su transformación a una base de datos simbólicos.

Un primer análisis consiste en trabajar con la base de individuos y a partir de ella considerar al hogar como objeto simbólico a fin de describir las variables simbólicas de interés.

Table 1. Variables Simbólicas

Variable de Intervalo	Variable Modal
Edad	Relación de Parent.
Ingreso Total Fliar(ITF)	Estado
Ingreso Per Cápita Fliar (IPCF)	Estado Civil
Monto de Ingreso por Act.	Nivel de Educ.
Principal(MIAP)	Categoría de Act.
	Categoría de Inac.

La obtención de los objetos simbólicos (OS) se ha realizado mediante el software SODAS (Symbolic Official Data Analysis System). Este software provee muy buenas posibilidades de aplicación para la manipulación de bases de datos de estadísticas oficiales. A continuación se muestran e interpretan algunos resultados, salidas del software SODAS.

Variables simbólicas de Intervalo

Table 2. Distribución de las Variable Simbólica Edad

limits:0 - 94	class width: 9.4
class1	0.0456
class2	0.0970
class3	0.1359
class4	0.1755
class5	0.1369
class6	0.1076
class7	0.1100
class8	0.1237
class9	0.0566
class10	0.0112
Central tend.: 42.87	Disp.: 21.36

De acuerdo a la Tabla 2, se observa que la mayoría de los hogares tiene ocupantes con edades comprendidas entre 19 y 47 años. Es decir, se destacan hogares con ocupantes jóvenes. En menor proporción, hogares con adultos mayores entre 68 y 75 años. Hay pocos hogares con ancianos de más de 75 años. La edad promedio es de 42 años, con una dispersión de 21 años.

La distribución de ITF es muy asimétrica positiva, hay muchos hogares con ingresos por debajo de la media. La mayoría de los hogares tienen ingresos totales inferiores a $25665, siendo el intervalo modal el correspondiente a ingresos que varían de $9000 a $17300, aproximadamente. Los hogares con ingresos totales superiores a $33970 no llegan al 9%. El ingreso medio, de aproximadamente $16746 , no es representativo (ver Tabla 3).

Table 3. Distribución de las Variable Simbólica ITF

limits: 750 - 83803	class width: 8305.3
class1	0.2807
class2	0.3779
class3	0.1575
class4	0.1007
class5	0.0350
class6	0.0197
class7	0.0197
class8	0.0000
class9	0.0044
class10	0.0044
Central tend.:16746.35	Disp.:12531.02

Table 4. Distribución de las Variable Simbólica IPCF

limits:187.5 - 28500	class width:2831.2
class1	0.3001
class2	0.2973
class3	0.1772
class4	0.1094
class5	0.0503
class6	0.0284
class7	0.0241
class8	0.0044
class9	0.0044
class10	0.0044
Central tend.6056.70	Disp.:4693.85

Table 5. Distribución de las Variable Simbólica MIAP

limits:0 - 60000	class width:6000
class1	0.3854
class2	0.3843
class3	0.1491
class4	0.0726
class5	0.0057
class6	0.0008
class7	0.0008
class8	0.0006
class9	0.0003
class10	0.0003
Central tend.8651.40	Disp.: 5895.57

La distribución de IPCF muestra un comportamiento similar a ITF, hay muchos hogares con ingresos por debajo de la media de $6056. El 88,4% de los hogares tienen ingresos per cápita inferiores a $11500, aproximadamente, siendo los intervalos modales los correspondiente a ingresos per cápita menores a $5850, aproximadamente. Hay alrededor de un 10% de hogares con IPCF superiores a $13100 (Tabla 4).

En cuanto a la variable Monto de Ingreso de la Actividad Principal (sólo para hogares con integrantes ocupados) presenta mucha dispersión debido a valores de ingresos altos muy atípicos. Aproximadamente el 92 % de los hogares tiene ingreso de la actividad principal de sus integrantes inferiores a $ 18000 (Tabla 5).

Variables Modales

Las variables analizadas son:

- Estado Civil (CH07) 1 = unido 2 = casado 3 = separado/a ó divorciado/a 4 = viudo/a 5 = soltero/a
- Estado (Social) 1=Ocupado 2=Desocupado 3=Inactivo 4= Menor de 10 años
- Categoría Ocupacional (CAT-OCUP)(Para ocupados y desocupados con ocupación anterior) 1 = Patrón 2 = Cuenta propia 3 = Obrero o empleado, 4 = Trabajador familiar sin remuneración 9 = Ns./Nr. NA=No aplicable
- Categoría de Inactividad (CAT-INAC) 1 = Jubilado/ Pensionado 2 = Rentista 3 = Estudiante 4 = Ama de casa 5 = Menor de 6 años. 6 = Discapacitado 7 = Otros NA=No aplicable

Del análisis de la salida de SODAS, para el caso de variables modales, se observó que, en cuanto al estado civil, el 43% de hogares tienen integrantes solteros, mientras que un 27% aproximadamente tiene integrantes casados, el resto pertenecen a otra categoría de Estado Civil (Tabla 6).

Aproximadamente el 36% de hogares tiene integrantes ocupados y el 52% de hogares, integrantes inactivos. Hay un 28% de hogares donde la ocupación de sus integrantes es obrero o empleado, un 17% de hogares con integrantes inactivos estudiantes y un 29% de hogares con integrantes jubilados (Tabla 7).

Table 6. Capacidades de las Variables Modales CH07 y Estado

Variable CH07		Variable Estado	
Modalidad	Media	Modalidad	Media
1	0.1121	1	0.3606
2	0.2721	2	0.0162
3	0.0702	3	0.5235
4	0.1171	4	0.0997
5	0.4285		

Table 7. Capacidades de las Variables Modales Cat-Ocup y Cat-Inac

Variable Cat-Ocup		Variable Cat-Inac	
Modalidad	Media	Modalidad	Media
1	0.0150	1	0.2866
2	0.0714	2	0
3	0.2834	3	0.1670
4	0.0022	4	0.0788
9	0.0047	5	0.0613
NA	0.6232	6	0.0050
		7	0.0245
		NA	0.3768

Gráficos Comparativos

Con el fin de comparar el comportamiento de algunas variables, en relación con la situación laboral del encuestado, se las grafica para los grupos de Ocupados, Desocupados e Inactivos. Esta representación se efectúa considerando como objetos simbólicos las categorías de la variable Estado de Ocupación. Es decir, se trabaja sólo con cuatro objetos: Ocupados, Desocupados, Inactivos y No Aplicables (Niños menores de 10 aõs). La semántica utilizada es la de probabilidades basadas en la frecuencia.

Las variables analizadas, además de CH07, Estado, Cat-Ocup. y Cat-Inac son:

- Relación de Parentesco (CH03) 1 = Jefe/a 2 = Cónyuge/Pareja 3 = Hijo/Hijastro/a 4 = Yerno/Nuera 5 = Nieto/a 6 = Madre/Padre 7 = Suegro/a 8 = Hermano/a 9 = Otros Familiares 10 = No Familiares
- Sexo (CH04) 1 = varón 2 = mujer
- ¿Sabe leer y escribir? (CH09) 1 = Si 2 = No 3 = Menor de 2 años
- ¿Asiste o asistió a algún establecimiento educativo?(colegio, escuela, universidad)(CH10) 1 = Si, asiste 2 = No asiste, pero asistió 3 = Nunca asistió
- ¿Finalizó ese nivel?(CH13) 1 = Si 2 = No 9 = Ns./Nr.
- Nivel Educativo(NIVEL-ED) 1 = Primaria Incompleta(incluye educación especial) 2 = Primaria Completa 3 = Secundaria Incompleta 4 = Secundaria Completa 5 = Superior Universitaria Incompleta 6 = Superior Universitaria Completa 7 = Sin instrucción 9 = Ns./ Nr.
- ¿Cuánto hace que está buscando trabajo?(PP10A)(Sólo Desocupados) 1 = ...menos de 1 mes? 2 = ...de 1 a 3 meses? 3 = ...más de 3 a 6 meses? 4 = ...más de 6 a 12 meses? 5 = ...más de 1 año
- Ha trabajado alguna vez?(PP10D)(Sólo Desocupados) 1= Si 2= No
- Edad (CH06CAT): la variable Edad se categoriza en clases de amplitud 10 años.
- Ingreso Total Familiar Categorizado (ITF-CAT): la variable ITF se categoriza en clases de amplitud $10000.

La visualización de un objeto simbólico se hace mediante un gráfico llamado Zoom Star. Esta representación se basa en los diagramas de Kiviat donde cada eje representa una variable. En el mismo gráfico pueden representarse variables categóricas, de intervalo, con pesos, taxonomías, etc, sin sobrecargar el gráfico. SODAS permite dos tipos de representación, en $2D$ y $3D$, que muestran diferentes niveles de detalle. La representación en $2D$ permite una impresión global del objeto, mientras que la representación en $3D$ nos da información más detallada. En $2D$ los ejes están unidos por una línea que conecta los valores más frecuentes de cada variable. Si hubiera un empate del valor más frecuente en varias modalidades, la línea uniría las dos. Cuando existe una variable intervalo la línea se une a los límites mínimo y máximo y el área entera se rellena [10]. Mostramos la visualización $2D$ de los OS de interés:

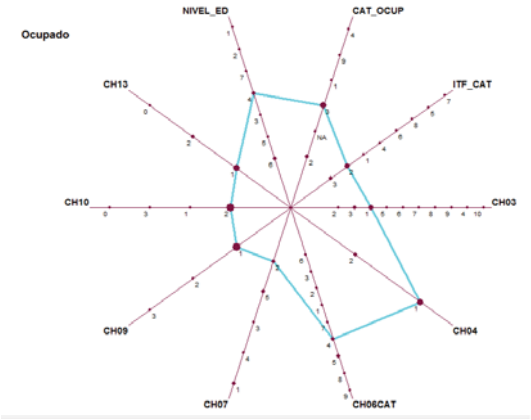


Fig. 1. Ocupados

El grupo de encuestados ocupados, Fig.1, se caracteriza (modos de cada variable) por haber asistido a un establecimiento educativo y finalizado ese nivel, teniendo un nivel de estudios secundario completo y, en menor proporción, universitario completo; su categoría ocupacional es obrero o empleado teniendo ingreso total familiar entre $10000 y $20000 y en menor proporción entre $20000 y $30000. Son jefes de hogar, varones, con edades entre 29 y 49 años, casados o solteros, casi en la misma proporción y todos saben leer y escribir.

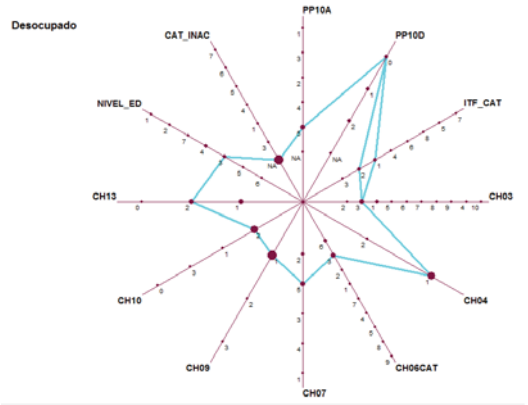


Fig. 2. Desocupados

El grupo de encuestados desocupados, Fig.2, se caracteriza por haber asistido a un establecimiento educativo y no haber finalizado ese nivel. Tienen un nivel de estudio secundario incompleto, han trabajado alguna vez y buscan trabajo hace más de un año, el ingreso total familiar de su hogar es menor a $20000. Son hijos, varones, con edades entre 29 y 39 años, solteros y todos saben leer y escribir.

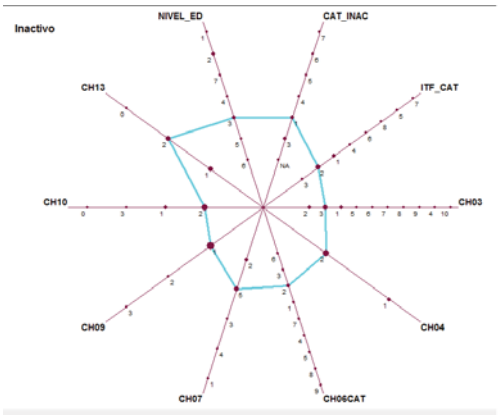


Fig. 3. Inactivos

El grupo de encuestados inactivos, Fig.3, se caracteriza por haber asistido a un establecimiento educativo y no haber finalizado ese nivel, teniendo un nivel de estudios secundario incompleto (el 35% son estudiantes y el 38% jubilados o pensionados). Tienen un ingreso total familiar inferior a $20000. Son hijos y en menor proporción jefes de hogar, varones, con edades entre 10 y 19 años, solteros, saben leer y escribir.

4 Conclusiones

Este trabajo muestra que los Datos Simbólicos son una herramienta de gran utilidad para el manejo y análisis de grandes volúmenes de datos.
Se destacan la posibilidad de trabajar con bases individuales relacionadas a fin de responder a diferentes propósitos de estudio, como también la generación de objetos simbólicos que son descriptos por variables simples, multivaluadas, probabilísticas y del tipo intervalo. Además, la teoría de objetos simbólicos posibilita trabajar con una población de interés específico. Por ejemplo, los integrantes del hogar de acuerdo a su Estado Social, permitiendo la comparación del comportamiento de diferentes variables en distintos grupos, donde cada grupo es un objeto simbólico de la tabla de datos simbólicos. También se destaca la posibilidad de la visualización exploratoria de los datos a través de gráficos $2D$ (o $3D$).

Existen numerosos análisis que pueden llevarse a cabo. En particular para la base analizada se pueden

construir objetos simbólicos tipo aserción obtenidos a partir de variables categóricas, por ejemplo combinando las modalidades de Nivel de Educación, Edad y Sexo. Para éstos objetos simbólicos se pueden aplicar métodos de clasificación no supervisada , a fin de detectar grupos homogéneos de integrantes de los hogares encuestados; respecto a variables de interés.

References

[1] Diday, E. (1987):Introduction a lapproche symbolique en analyse des données. Premiéres Journées Symbolique - Numérique. CEREMADE, Université Paris Dauphine, 21-56.

[2] Diday, E. (1988): The symbolic approach in clustering and related methods of data analysis : the basic choices. In: H.H. Bock (Ed.)

[3] Diday E. (1991): Des objets de lAnalyse des Données á ceux de lAnalyse des Connaissances in Induction symbolique et numérique. Y. Kodratoff and E. Diday edit. CEPADUESEDITIONS, Toulouse, France

[4] Diday E: (1995):Probabilist, possibilist and belief objects for knowledge analysis. Annals of Operations Researche 55, pp. 227-276.

[5] Billard, L. and Diday, E. (2003). From the statistics of data to the statistics of knowledge: Symbolic Data Analysis. Journal of the American Statistical Association, 98 (462), pp. 470487.

[6] Billard, L., Diday, E. (2007): Symbolic Data Analysis: Conceptual Statistics and Data Mining. Wiley.

[7] Brito, P. (2014). Symbolic Data Analysis: another look at the interaction of Data Mining and Statistics. WIREs Data Mining and Knowledge Discovery, Volume 4, Issue 4, July/August 2014, 281-295.

[8] Diday, E. An Introduction to Symbolic Data Analysis and the Sodas Software. University Paris,Dauphine.(2000)

[9] Diday, E. The state of the art in symbolic data analysis: overview and future. (2008)

[10] Bock, H.-H.; Diday, E. (2000): Analysis of Symbolic Data: Exploratory methods for extracting statistical information from complex data. Berlin-Heidelberg: Springer-Verlag.

[11] Bravo Llatas, M. Análisis de Segmentación en el análisis de datos Simbólicos.Tesis de la Universidad Complutense de Madrid, Facultad de Ciencias Matemáticas, Departamento de Estadística e Investigación Operativa. (2001)

Análisis de las Topologías IoT en Entornos Fog Computing mediante simulación

Javier Sillero Ros[1], Nelson Rodríguez[1], Matías Montiveros[1], Maria Murazzo[1], Fabiana Píccoli[2], Miguel Méndez Garabetti[3],

[1]*Departamento de Informática, F. C. E. F. y N, Universidad Nacional de San Juan*
javiersilleroros@gmail.com, nelson@iinfo.unsj.edu.ar, nicolasmontivero.nw273@gmail.com, marite@unsj-cuim.edu.ar
[2] *Departamento de Informática - F.C.F.M. y N, Universidad Nacional de San Luis*
mpiccoli@unsl.edu.ar
[3]*Instituto de Investigaciones, Facultad de Informática y Diseño, Universidad Champagnat*
mendez-garabettimiguel@uch.edu.ar

Abstract

El modelo Cloud Computing está demostrando algunas debilidades con el surgimiento del IoT, la gran cantidad y variedad de datos que se generan y envían a la nube está saturando las redes y surgen problemas como: alta latencia, baja disponibilidad de ancho de banda, interrupciones momentáneas de Internet y diversos temas de seguridad. Es por ello que surge el modelo Fog Computing, con el objetivo de llevar parte del procesamiento, control y almacenamiento al extremo, específicamente sobre los dispositivos de red.

La variedad de topologías de red impacta sobre los parámetros de la continuidad IoT – Cloud. Se desconoce qué tipos de redes emplear a la hora de implantar modelos Fog y dónde ubicar el procesamiento y otras funcionalidades, sumado a la heterogeneidad de dispositivos y protocolos de comunicación en el extremo IoT. Es por eso que, para llevar a cabo una evaluación de rendimiento, se debe utilizar la simulación de entornos Fog. Este trabajo analiza las topologías IoT más conocidas mediante el empleo de técnicas de simulación y presenta una aproximación acerca de qué tipos de modelos topológicos vuelven del Fog Computing un modelo más eficiente.

Palabras claves: Edge Computing, Fog Computing, Fog Torch Pi, IoT, simulation, Topology IoT.

1. Introducción

Cloud Computing es un modelo tecnológico que permite el acceso ubicuo, adaptado y bajo demanda en red a un conjunto compartido de recursos de computación configurables (por ejemplo: redes, servidores, almacenamiento, aplicaciones y servicios), que pueden ser rápidamente provisionados y liberados con un esfuerzo de gestión reducido o interacción mínima con el proveedor del servicio [1].

La expansión del Cloud sumado a los requerimientos de los usuarios ha terminado de consolidar este modelo. A la Infraestructura, Plataforma y Software como servicio, se le agregaron: Redes, Almacenamiento, Contenedores, Desktop, Base de datos, Seguridad, Video y Comunicaciones Unificadas como Servicio, entre otras. En definitiva, lo que actualmente se conoce como XaaS o Everything as a Service [2].

Por otro lado, Internet de las cosas (IoT) es una tendencia reciente de la computación distribuida que integra aspectos de la vida real a escalas masivas. En 1999, Kevin Ashton del Instituto Tecnológico de Massachusetts (MIT) acuñó el término Internet de las Cosas. En ese momento, las tecnologías de automatización comenzaban a pasar de la fábrica a nuevos entornos como hospitales, bancos y oficinas.

A medida que las primeras implementaciones de M2M (machine-to-machine) se hicieron más sofisticadas, las máquinas comenzaron a ser conectadas a otros tipos de dispositivos como servidores, y estos servidores se trasladaron a centros de datos y al Cloud. Hoy en día, Internet de las Cosas puede incluir productos industriales y comerciales, productos cotidianos como lavavajillas y termostatos, y redes locales de sensores para vigilar granjas y ciudades [3].

Las soluciones que ofrece IoT promueven la incorporación de todos los dispositivos a la red y se pronostica que entre 20 y 50 millones de los mismos se añadirán a Internet para 2020, creando una economía de más de 3 billones de dólares [4]. En consecuencia, 43 billones de gigabytes de datos serán generados y necesitarán ser procesados en los centros de datos del Cloud. Las aplicaciones que generan datos en dispositivos de usuario, como teléfonos inteligentes, tablets y dispositivos portátiles, usan actualmente el Cloud como un servidor centralizado, pero pronto se convertirá en un modelo informático insostenible [5].

La importancia de IoT y las capacidades del Cloud, imponen que esta asociación sea por demás necesaria

y ofrece muchas ventajas. La misma debe darse de forma eficiente y efectiva, con tráfico fluido de las cosas al Cloud y viceversa, lo que se denomina continuidad IoT - Cloud. Sin embargo, debido a la gran cantidad de dispositivos IoT con plataformas heterogéneas, el desarrollo de las nuevas aplicaciones es una tarea un tanto difícil [6].

En la actualidad, el modelo de Cloud Computing está encontrando serias dificultades para satisfacer los requerimientos de Internet de las Cosas. Por ejemplo, para monitoreo de salud, respuestas de emergencias y otras aplicaciones embebidas sensibles a la latencia, el retraso causado por la transmisión de datos al Cloud y el retorno de los mismos es inaceptable.

Por lo tanto, hay muchos datos que necesitan ser atendidos inmediatamente para las operaciones 24/7, y siempre en tiempo real. Los antiguos enfoques ya no funcionan, se necesita cambiar la arquitectura, la metodología, la topología y el pensamiento que sustenta la ciencia de datos. Se debe dejar atrás la antigua arquitectura centralizada y pasar a un modelo federado, que está compuesto de modelos distribuidos y centralizados que funcionan en armonía.

Para ayudar a abordar estos problemas, se ha propuesto el concepto de Fog Computing y Edge Computing. De acuerdo con este paradigma, los recursos informáticos están disponibles en el borde de la red, cerca de (o incluso ubicado junto con) los dispositivos finales [7].

Colocando los recursos informáticos en las proximidades de los dispositivos que generan los datos se reduce la latencia de comunicación. Además, los datos intensivos en red pueden procesarse y analizarse a un solo salto de los dispositivos finales, reduciendo así las demandas de ancho de banda en los enlaces de red a dispositivos distantes de los centros de datos [8].

La integración de Fog Computing con IoT crea una nueva oportunidad para los servicios, que se llama Fog como Servicio (FaaS) [9]. FaaS habilitará nuevos modelos de negocios para entregar servicios a los clientes. A diferencia del Cloud, que es operado principalmente por grandes compañías que construyen y operan enormes centros de datos, FaaS permitirá a pequeñas y grandes empresas implementar y operar servicios de computación, almacenamiento y control, privados o públicos y a diferentes escalas, para satisfacer las necesidades de una amplia variedad de clientes [10].

El presente trabajo analiza las topologías más utilizadas en IoT, a las cuales se las embebe en entornos Fog Computing bajo la simulación de las mismas. El esquema que respeta el trabajo es el siguiente: en la Sección 2 se presenta la definición del modelo Fog Computing y sus implicancias, en la Sección 3 se describen los desafíos que debe resolver esta nueva arquitectura, mientras que en la Sección 4 se presentan los objetivos del trabajo para, luego, en la Sección 5, aclarar cuál es la problemática actual y continuar explicando la metodología empleada en el estudio en la Sección 6. La Sección 7 describe las topologías usadas en IoT, en tanto que en la Sección 8 se describen los detalles del proceso de simulación y en la Sección 9 las conclusiones obtenidas del mismo. Por último, en la Sección 10, se mencionan los trabajos a futuro.

2. Fog Computing

Llevar parte de las funciones de los centros de datos al extremo de la red ha sido puesto en práctica anteriormente, entre sus antecedentes se puede nombrar a las Redes de Distribución de Contenido. Las mismas surgen a fines de la década del 1990, cuando Akamai introduce las Redes de Entrega de Contenido (CDN) para acelerar el rendimiento web. Una CDN utiliza nodos en el borde cerca de los usuarios para captar previamente y almacenar en caché el contenido web.

Las CDN son valiosas para el contenido de video, por el ahorro de ancho de banda debido al almacenamiento en caché. Pero el nuevo modelo Edge Computing generaliza y amplía el concepto de CDN al aprovechar la infraestructura de computación en la nube [11].

En 2001 se generalizó otro concepto, introduciéndose el término Cyber Foraging para la amplificación de las capacidades informáticas de un dispositivo móvil mediante el aprovechamiento de la infraestructura cercana [12]. Esto permite orientar el futuro de las investigaciones acerca del procesamiento fuera del Cloud.

En 2009 se desarrolla el concepto de Cloudlet, que en la actualidad es mantenido por la Universidad de Carnegie Mellon y varios sponsors como Intel, Vodafone e IBM, entre otros. Se define al Cloudlet como un nuevo elemento arquitectónico que surge de la convergencia de la informática móvil / IoT y el Cloud Computing. Representa el nivel medio de una jerarquía de 3 niveles: dispositivo móvil o IoT - cloudlet - Cloud. Un cloudlet se puede ver como un "*centro de datos en una caja*" cuyo objetivo es "*acercar el Cloud*" [13].

En 2012, Flavio Bonomi y colaboradores presentaron el término Fog Computing (informática en la niebla) para referirse a una dispersa infraestructura de Cloud [14]. Sin embargo, en este caso, la motivación para la descentralización es la escalabilidad de la infraestructura IoT en lugar del rendimiento interactivo de las aplicaciones móviles.

Fog es una arquitectura emergente para almacenamiento, control, cómputo y red, que distribuye estos servicios cerca del usuario final a lo

largo de la continuidad, desde el Cloud a las cosas (dispositivos finales). No resulta un simple traslado de la funcionalidad del Cloud a los nodos intermedios, dado que las aplicaciones IoT son diferentes de las basadas en Web. Ciertas aplicaciones IoT necesitan respuestas en tiempo real o con un límite de tiempo conocido y además muchas de estas presentan un gran número de dispositivos conectados que pueden variar, resultando un desafío importante mantener la escalabilidad.

Los términos Edge y Fog suelen llevar significados diferentes según el autor que los referencie, lo cual genera confusión, ya que se suele definir Mist como el extremo final (en los dispositivos) y Fog como el extremo medio (dispositivos de interconexión por lo general) o, en otros casos, no discernir entre Fog y Mist Computing. Por ello es necesario indicar adecuadamente la definición que se va a utilizar de estos términos.

En particular, Fog y Mist Computing, usualmente se asocian a IoT en el extremo y ambos son incluidos como parte de Edge Computing. A su vez, se suele utilizar el término Edge cuando se los asocia a Mobile Cloud, o sea, en el extremo se dispone de computación móvil (smartphones).

En este estudio se considera a Fog Computing como aquella arquitectura que se ubica solamente en los dispositivos de red (a un nivel intermedio), mientras que a Mist Computing como aquella que se refiere a los dispositivos finales. El término Edge Computing es quien engloba a los anteriores.

3. Desafíos Fog Computing

La computación en el borde (Edge Computing) está recién en sus comienzos, tal es así que los investigadores suponen que para su real consolidación se necesitará de la construcción de nuevos estándares y desarrollos.

A pesar de presentar notables ventajas con respecto al modelo centralizado, mantener la continuidad de forma eficiente y efectiva no se puede obtener simplemente agrupando el Cloud con el extremo. Los sistemas IoT necesitan interactuar con otros, con el Cloud y con diversos tipos de usuarios y dispositivos en un ambiente distribuido y heterogéneo.

No sólo resulta necesaria una redistribución del cómputo, las funciones de red, el almacenamiento y el monitoreo, sino que, además, dicha reubicación debe ser estratégica para impedir que el sistema colapse o presente demoras inadecuadas.

Los modelos de programación concurrentes utilizados actualmente no son adecuados para aplicar en la problemática presentada. La programación paralela con memoria compartida como OpenMP o Cuda no puede ser aplicada. El modelo de memoria distribuida como MPI tampoco puede ser aplicado dado que se diseñó para otro tipo de arquitecturas y además no puede ejecutarse en el extremo. El modelo de threads solo es aplicable dentro de un mismo equipo. Por lo tanto, se deben desarrollar soluciones basadas en el modelo de programación de actor o alguno similar que permita la concurrencia basada en memoria distribuida, pero aplicable al Edge [15].

Resultará necesario también utilizar eficientemente los recursos, lograr un adecuado balanceo de carga para la distribución entre el borde y el Cloud, definir las API's y la administración y compartición de servicios y las comunicaciones en redes definidas por el software y virtualizadas [16].

El problema de la orquestación de la variedad de servicios complejos es un importante desafío a resolver, que resulta muy complicado debido a los ambientes altamente dinámicos, a la cantidad de aplicaciones instaladas y dispuestas para Fog y a la necesidad de soportar diferentes dominios de aplicación para adaptar el servicio a la heterogeneidad extrema de la infraestructura. Por cada instancia las aplicaciones Fog no pueden siempre contar con la disponibilidad de los dispositivos de computación más poderosos, quienes ejecutan algoritmos de orquestación complejos, por lo que, en condiciones críticas (por ejemplo, cuando la infraestructura se cae por algún hecho climático), la infraestructura Edge debe ser capaz de orquestar los servicios aún en presencia de computación limitada con un grado intrínseco de resiliencia.

Otro desafío importante es la capacidad de crear aplicaciones auto adaptativas, las cuales deben ser capaces de adaptar su comportamiento basado en el ambiente subyacente, por ejemplo, en algunos casos los servicios requeridos (por ej. alta capacidad de almacenamiento o sensores de alta precisión) no pueden ser alcanzados sin dejar de ofrecer el servicio que el usuario espera, aunque con cierta degradación.

En la actualidad se está en la antesala de la cuarta ola industrial, la misma emplea tecnologías como la inteligencia artificial, la computación cognitiva, el Cloud, la analítica e Internet de las Cosas para realizar el trabajo de una manera totalmente automatizada y en constante aprendizaje. Es la primera ola que no necesita a los humanos para realizar llevarse a cabo [17].

4. Objetivo

El presente trabajo tiene como fin encontrar aquellos tipos de configuraciones topológicas que vuelven al modelo Fog Computing aplicable en entornos genéricos mediante la utilización de herramientas de simulación de entornos Fog. El interés está centrado en hallar configuraciones de red (topologías) que soporten las características propias de IoT, de manera que el comportamiento/performance de las mismas justifique el empleo de un modelo Fog Computing y

exponga sus ventajas respecto del despliegue centralizado del Cloud tradicional.

5. Situación actual y problemática

Si bien el modelo de Cloud Computing ofrece una gran capacidad de almacenamiento, control y cómputo, el mismo no es adecuado para complementarse con IoT. Hoy por hoy las redes de conexión a Internet están cada vez más congestionadas por la cantidad de datos que fluyen desde los dispositivos finales hacia el corazón de la red: los servidores del Cloud. Es por ello que surgen problemáticas relacionadas con la alta latencia, la baja disponibilidad de ancho de banda, la pérdida de información, y, por ende, un menor throughput del sistema en general. Por otro lado, la cantidad de dispositivos IoT es cada vez más grande, así como la cantidad de información que se enviará hacia el Cloud en un futuro.

Los modelos que soporten IoT debieran priorizar, entre otras cosas, cuatro cuestiones básicas para un buen funcionamiento: la minimización de la latencia, la conservación del ancho de banda, el abordaje de cuestiones de seguridad y el movimiento de la información a los lugares más indicados para su procesamiento [21]. Dichas cuestiones serían abordables por el modelo del Cloud, pero la dependencia de las redes instaladas no permite una completa satisfacción de esos principios.

Fog Computing, por su parte, busca reducir el tráfico en la red, logrando una disminución en la latencia, y persigue un acercamiento del cliente con el servidor y una ampliación en el número de servidores, de modo que se aproveche de mejor manera el ancho de banda disponible, se incremente la productividad de las aplicaciones y servidores y se mejore la experiencia de usuario. En este nuevo paradigma, todo aquel dispositivo que tenga capacidad de procesamiento y se encuentre conectado a la red (siendo un dispositivo de red) es capaz de pre-procesar información y puede ser considerado un nodo Fog.

Planteado el nuevo modelo y una nueva idea de procesamiento surgen problemáticas tan o más preocupantes: ¿Cuándo es conveniente emplear un modelo Fog Computing que soporte la creciente cantidad de información que se genera?, ¿Cómo implantar el nuevo modelo, bajo qué criterios?, ¿La información seguirá circulando al Cloud o bastará con el procesamiento en el borde?, ¿Cómo generar un correcto balanceo de cargas si el Cloud continúa teniendo participación y las redes son el intermediario entre estos?, ¿Qué tipo de tareas realizar en el Cloud y cuales realizar en los nuevos nodos Fog?.

Algunos de los interrogantes tienen respuesta desde la teoría de Fog Computing, partiendo de que el Cloud y el Fog son complementarios y las cargas se verán repartidas entre ambos modelos, pero muchos otros generan campos de investigación que están empezando a trabajarse a nivel mundial. La línea adoptada por los autores del presente trabajo es la simulación de topologías IoT embebidas en entornos Fog Computing

El desafío de esta investigación está en evaluar los diferentes tipos de topologías más utilizadas en IoT y determinar bajo qué condiciones las aplicaciones distribuidas en las mismas funcionarán de manera eficiente empleando un modelo Fog. Realizar las pruebas correspondientes para hallar resultados al respecto llevaría mucho tiempo y requerirían de bastante inversión, sin tener la certeza de que el modelo propuesto funcionará, es por ello que, la simulación, es una solución alternativa para investigar situaciones como estas.

6. Metodología empleada

Para llevar a cabo los estudios de laboratorio pertinentes a las topologías IoT más utilizadas se empleó una metodología cuantitativa en el marco de una investigación experimental, de manera que se manipularon variables explícitas del área a través del empleo de herramientas de simulación.

Dicha simulación se realiza en software especialmente diseñado para ello (simuladores de entornos Fog Computing), del cual se obtienen datos de Calidad de Servicio (QoS) de la red simulada, utilización de recursos hardware y ubicación de las aplicaciones, y se evalúan luego los comportamientos observados bajo los distintos patrones de configuración ingresados a través de la topología y las aplicaciones a correr. De esta manera, la creación de topologías de diversa variedad y tamaño permitirá definir bajo qué criterios estructurales es más conveniente emplear el modelo Fog.

Utilizando dichas técnicas de estudio, el grupo de investigación se encuentra trabajando para llevar a cabo evaluaciones más profundas acerca del comportamiento de las diferentes topologías de red en entornos Fog Computing, considerando la escalabilidad de las redes como el primer aspecto a hacer variar en las próximas simulaciones.

De esta manera, mediante la simulación de entornos Fog se pretende evaluar el rendimiento de cada una de las configuraciones topológicas, haciendo variar, además de su escala, aspectos como el tipo de conectividad, el tipo y la cantidad de los dispositivos que componen las redes, y las aplicaciones a desplegar en los mismos (considerando las políticas de ejecución, las cargas de procesamiento asignadas y el tamaño de las mismas). Todo ello, con el objeto de hallar conclusiones que permitan determinar las distribuciones topológicas de IoT que presentan en

general una mejor performance en entornos Fog Computing.

7. Topologías IoT

Recabando opiniones aportadas por expertos en redes IoT, especialmente relacionados los mismos con el sector empresarial, se observa que existen inicialmente 3 modelos topológicos básicos para IoT, los mismos son: el modelo Estrella, el modelo Peer-To-Peer y el modelo Mesh.

Para cada configuración topológica, se denominará como "nodo" a todo aquel dispositivo de red que forme conforme la misma. Se llamará "extremo superior" a los recursos de Cloud Computing disponibles para realizar las tareas de procesamiento o almacenamiento (siempre ubicados sobre nubes de procesamiento públicas), mientras que el "extremo inferior" serán los dispositivos finales de la red, es decir, los dispositivos a través de los cuales se obtiene la información y se les debe devolver la mismas ya procesada.

Estructuralmente el modelo Estrella se basa en la centralización del cómputo, del almacenamiento y del control. En una red IoT de tipo Estrella, un dispositivo final está conectado únicamente a un solo nodo de la red, cumpliendo este un papel de nodo central para con el mismo. A su vez, cada uno de estos nodos de red está conectado a un nodo servidor, el cual podría ser una conexión con el Cloud o con otro nodo de la red con capacidades superiores o funcionalidades diferentes, cumpliendo este último un papel de nodo central de mayor jerarquía. De esta manera, para que dos dispositivos finales puedan comunicarse entre ellos, la información debe viajar hasta el nodo central que comunica los caminos hacia ambos dispositivos y, luego, completar el camino desde este hacia el destinatario.

Este tipo de estructura de red presenta ventajas en un ambiente IoT debido a su fácil configuración, refiriéndose a la misma como al agregado y la remoción de dispositivos finales y a la detección de fallos. Además, dicha configuración, cargada generalmente sobre un dispositivo central que se encarga de solucionar todas aquellas situaciones complejas, permite que la performance de la red sea consistente, predecible y buena -de baja latencia y suficiente ancho de banda-. La latencia, motivo que preocupa a las aplicaciones IoT, se vería decrementada con el uso de este tipo de topologías ya que se reducen la cantidad de saltos o hops que son necesarios para transportar la información a destino.

A pesar de todo ello, las redes Estrella presentan serias desventajas que afectan a la performance en ambientes IoT. La centralización de las funcionalidades soluciona muchos problemas, pero presenta una de las desventajas más serias: un único punto de fallo. Si el dispositivo central con el cual se comunican los diferentes nodos de la red falla, el funcionamiento de la red se ve deteriorado por completo. Por el contrario, si fallan los nodos más cercanos a los dispositivos finales -hablando de una red estrella en forma jerárquica- el sector de la red que se encuentra en fallo puede ser aislado rápidamente, mientras que el resto funciona de modo correcto.

Las redes de tipo Estrella también suelen enfrentar dificultades con respecto a las radio-interferencias, el rango de transmisión (limitado al rango de transmisión del dispositivo que cumpla la función de nodo central), y el consumo de energía, el cual depende de la distancia existente entre los dispositivos conectados, para lo que hay que recordar que los dispositivos IoT funcionan mayoritariamente alimentados por baterías.

En cuanto a las redes de tipo Peer-to-Peer, las mismas presentan una estructura en la cual todos los nodos existentes en la red están conectados entre sí, es decir, existe un link permanente entre cada nodo que conforme la red. De esta manera, este tipo de configuraciones permiten una conexión adaptativa a la situación en la cual se encuentra la red respecto al tráfico de información.

Las redes de pares (Peer-to-Peer) no son tan sencillas de configurar, debido a que es necesario establecer los links entre cada uno de los nodos de red agregados y los ya existentes y, luego, conectar los dispositivos finales con los mismos. De este modo, el extremo inferior de la red posee conexión entre sí y con el extremo superior a través de los nodos de red más cercanos (a los cuales están conectados). Hoy por hoy existen configuraciones como estas que permiten la comunicación y la distribución de archivos entre hosts pertenecientes a la misma red, mediante el establecimiento de links directos entre los mismos.

Además de todo ello, las redes de pares presentan varias desventajas, entre las cuales se destacan el rango de comunicación limitado al salto más cercano y, lo más importante en términos de opinión de especialistas que trabajan en el mercado, las redes Peer-to-Peer no son muy útiles para ambientes IoT, debido a que es muy extraño e innecesario que dos dispositivos se comuniquen directamente entre ellos (links redundantes y mayores problemas de interferencia y seguridad entre los mismos).

En un ambiente Fog Computing, las redes Peer-to-Peer serían representadas de manera que los nodos de red sean los que se comunican entre sí, mientras que los dispositivos finales se conectan a estos. Así, la latencia se vería disminuida en caso de que los nodos pre-procesen parte de la información y puedan retornar resultados a los actuadores correspondientes.

Por último, las redes Mesh, las cuales son una combinación de los otros dos tipos, consisten

básicamente en tres clases de nodos: los gateways o Dispositivos de Funcionalidad Completa (FFD), los sensores/actuadores-routers o Dispositivos de Funcionalidad Reducida (RFD) [18], y los sensores/actuadores simples. Los Dispositivos de Funcionalidad Completa representarán, en un ambiente Fog, la salida al cloud o aquellos servidores a los cuales los nodos de red se conectan para llevar la información, mientras que los Dispositivos de Funcionalidad Reducida son los nodos de red que poseen capacidades de procesamiento y pueden correr algún tipo de aplicación que cumpla determinada función. Los sensores o actuadores simples son los dispositivos finales, considerados fuente de datos y destinatarios de información ya procesada.

Este tipo de topología combina en parte las ventajas de los modelos anteriores: son redes adaptativas, es decir, continúan funcionando cuando los links caen e incluso cuando ciertos nodos caen -lo que se denomina una mayor tolerancia a fallos-. El mantenimiento y la escalabilidad se tornan menos complicados, debido a que se subdivide la red con diferentes configuraciones y aquellos sectores de estructura estrella son mucho más simples de adaptar y escalar. Las interferencias son más severas y la capacidad de retransmisión que otorga el modelo Peer-to-Peer da aún mayor flexibilidad y tolerancia a los fallos de conectividad.

Así como se combinan las ventajas de los modelos Estrella y Peer-to-Peer, las desventajas también se hacen presente en las redes Mesh producto de la herencia. La redundancia de los links por parte de la configuración Peer-to-Peer continúa acomplejando la configuración y el mantenimiento de gran parte de la estructura y, como si fuera poco, continúa existiendo la posibilidad de la conformación de un cuello de botella, es decir, la existencia de un único punto de fallo. A todo ello, se le suman algunos otros problemas como es el caso de la latencia frente a los fallos y la búsqueda de caminos alternativos para hacer circular la información.

La mayor parte de las opiniones en el área de IoT defienden a las redes de tipo Estrella como las mejores candidatas, es por ello que, teniendo en cuenta dichas estructuras topológicas, se realizaron los estudios a ser explicados en la sección siguiente.

8. Investigación y Resultados

8.1. Herramienta de Simulación

La herramienta de trabajo utilizada se denomina Fog Torch Pi, la misma es presentada como un prototipo en el proyecto de investigación: “Through the Fog”, actualmente en ejecución en la Universidad de Pisa (Italia), y publicada junto con el paper: “How to best deploy your Fog applications, probably “[19]. La misma está desarrollada en Java y se encuentra disponible para todo público en un repositorio de GitHub [https://github.com/di-unipisocc/FogTorchPI], desde el cual se la puede importar y modificar para cualquier tipo de uso, solo siendo necesario la mención a los creadores.

Existen tres ramas de la aplicación sobre Git al momento de la escritura del artículo, la rama Master, la rama Cost Model y la rama Multithreaded, que, al pertenecer a diferentes ramas y como sus nombres lo indican, permiten realizar diferentes actividades debido a las funcionalidades que cada una aporta. Para el trabajo se utilizó la versión Cost Model, la cual “*…permite expresar las capacidades de procesamiento y los atributos promedio de la Calidad de Servicio (QoS) de la infraestructura Fog, junto con los requerimientos de procesamiento y QoS de la aplicación, y determinar los despliegues de la aplicación sobre la infraestructura Fog que satisfacen todos los requisitos preestablecidos*” [19]. De esta manera, el proceso de simulación consta de tres fases: la definición de la infraestructura, la definición de la aplicación y la simulación propiamente dicha.

En la primera de las fases se determinan las topologías a poner a prueba, explicadas en la sección anterior, definiendo los nodos de las mismas (servidores Clouds, nodos Fog, dispositivos finales) en cuanto a sus características de hardware y de software. También se determinan los links entre los mismos, lo que incluye definir latencia, ancho de banda (subida y descarga por separado) y la probabilidad con la que ese ancho de banda está garantizado. La segunda fase consiste en crear la aplicación, para lo cual se definen los módulos de software que funcionarán y deberán ser mapeados de la mejor manera sobre la infraestructura, cumpliendo con todos los requisitos. Para realizar esto se precisan las estructuras de hardware y software requeridas por cada uno de los módulos de la aplicación. Estas dos primeras etapas se realizan sobre un archivo .json, que luego es interpretado por la herramienta.

Por último, se lanza la simulación, la cual, en pocas palabras, va a buscar todas las combinaciones de despliegue posibles de la aplicación generada sobre la topología señalada, devolviendo como resultado, para cada uno de los despliegues posibles, el porcentaje en que satisface los requisitos de QoS, el promedio de consumición de recursos de los dispositivos Fog (considerando almacenamiento y memoria RAM) y el costo monetario de montar una infraestructura de esas características. Dicho costo es calculado gracias a que se puede ingresar, a medida que se construye la infraestructura, los costos de utilización de los recursos, ya sean hardware, software, y también el costo de la utilización de los links de comunicación entre los nodos de la red.

Esta herramienta estima la QoS de los links usando una distribución probabilística, describiendo variaciones de latencia y ancho de banda. Dicho modelo de distribución se basa en el modelo de Monte Carlo y, el resultado entregado, que se titula como Garantía de Calidad de Servicio, representa el porcentaje de corridas en las que un despliegue en particular cumplió satisfactoriamente con todos los requisitos de QoS impuestos tanto por la infraestructura como por las aplicaciones a desplegar sobre la misma. De dicha manera, el mencionado porcentaje de QoS estima cuán bien un despliegue satisface la Calidad de Servicio en su conjunto.

8.2. Condiciones Generales de las Simulaciones

Previo al presente trabajo se realizaron una serie de simulaciones, publicadas en el artículo académico [22], que arrojaron resultados parciales acerca del comportamiento y la performance de los diferentes modelos topológicos más utilizados en IoT, pero todas ellas fueron a baja escala, con la consideración de pocos dispositivos finales que generan tráfico de información hacia pocos nodos Fog. Con ello se obtuvieron despliegues posibles para situaciones representativas de bajo tamaño, planteadas a la hora de realizar dichas simulaciones, de manera que las conclusiones tratan acerca del comportamiento de las redes propuestas y se expresan, al igual que los resultados del presente trabajo, mediante valores de QoS/performance, utilización de recursos y distribución de aplicaciones/cargas de trabajo.

Tomando como punto de partida dichas simulaciones, se decidió utilizar las mismas técnicas, pero esta vez a gran escala, variando la cantidad de dispositivos finales conectados, pero también incrementando la cantidad de nodos de red (nodos Fog) implantados en las topologías. Es por ello que se tomaron en cuenta nuevamente los 3 modelos topológicos más usados en IoT: modelo Estrella, modelo Peer-to-Peer y modelo Mesh.

Previo a la definición de las estructuras a simular, se determinaron los tipos de enlaces a emplear en dichas topologías. Los valores de los mismos son reales, con tecnologías que al día de hoy se encuentran vigentes y son utilizadas con frecuencia para establecer este tipo de redes IoT en el mercado.

En las diferentes topologías planteadas como casos de estudio, los enlaces entre los nodos Fog siempre son de tipo 3G, mientras que los links al extremo superior variaron entre WLAN y Satelital. La Tabla 1, extraída de [20], aclara cada una de las variables a considerar en la construcción de las topologías y el establecimiento de sus enlaces.

Tabla 1 – Perfiles de links considerados

Link	Latencia (ms)	Descarga (Mbps)	Subida (Mbps)
Satélite 14M	40	98%: 10,5 2%: 0	98%: 4,5 2%: 0
3G	54	99,6%: 9,61 0,4%: 0	99,6%: 2,89 0,4%: 0
4G	53	99,3%: 22,67 0,7%: 0	99,4%: 16,97 0,6%: 0
VDSL	60	60	60
Fibra Óptica	5	1000	1000
WLAN	15	90%: 32 10%: 16	90%: 32 10%: 16

Así como se determinaron las medidas de los enlaces, se definieron los costos y las capacidades de las diferentes instancias virtuales a utilizar en el Cloud, las mismas fueron abstraídas un tanto de la realidad, llevándolas a valores representativos que permitan la comprensión de lo que se está simulando. Además, se definieron los costos de utilización de los dispositivos finales, para los cuales se determinó un número fijo de invocaciones a los mismos dependiendo de su tipo: sensores o actuadores. Las Tablas 2 y 3 expresan dichos valores.

Tabla 2 – Características de Máquinas Virtuales

Tipo de MV	vCPUs	RAM (GB)	HDD (GB)	Costo (USD)
tiny	1	1	10	165
small	1	2	20	236
medium	2	4	40	1182
large	4	8	80	1774
xlarge	8	16	160	2365

Tabla 3 – Características de Dispositivos Finales

Tipo Disp.	Costo p/invocación	Cant. Invocaciones
Sensor	0,0002 USD	2880
Actuador A	0,0006 USD	96
Actuador B	0,001 USD	192

En cuanto a las cantidades de dispositivos finales se consideraron desde 40 hasta 240 dispositivos conectados a los nodos de red de la manera más representativa posible, es decir, en ocasiones se sobrecargaron nodos con conexiones del extremo inferior, mientras que, en la mayoría de los casos, se trabajó lo más balanceado posible. Cabe remarcar que la herramienta de simulación utilizada permite declarar los dispositivos finales que se conectan a cada nodo Fog y también explicitar a qué dispositivos del extremo inferior deben tener acceso los módulos de aplicaciones a desplegar en la topología. Respecto a la cantidad de nodos de Fog, se emplearon desde un

solo nodo hasta seis nodos Fog dependiendo del tipo de red y del objetivo de cada simulación.

Definido todo ello, se decidió tomar la misma distribución de aplicaciones que en las pruebas ya realizadas (la cual tienen como fuente el artículo [20]), a diferencia de que se permite la duplicación de alguno de los módulos para observar sus resultados. De este modo, la aplicación a desplegar en cualquier topología propuesta consiste de tres tipos de módulos de software: un módulo de almacenamiento (Storage), un módulo de visualización de resultados (Dashboard) y uno o más módulos de cómputo (IoTController). La Figura 1 ilustra la comunicación entre dichos módulos.

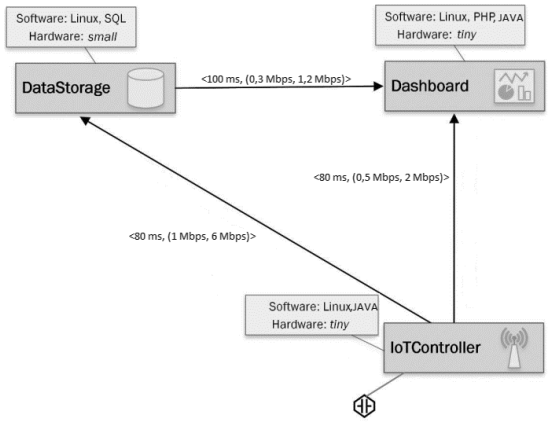


Figura 1 – Conectividad de Módulos de Software a desplegar

Por último, es necesario remarcar lo que concierne a la cantidad de simulaciones. Para cada uno de los casos de definió un conjunto de pruebas o archivos de entrada que representan una situación en particular, modificando en ocasiones una sola variable y en otras más de una. En promedio se realizaron 16 pruebas para cada tipo de topología, de las cuales, para cada prueba, la simulación consiste en 100000 corridas. El número de corridas no es elegido al azar, sino que se obtuvo en base a que, a menor cantidad de corridas, los resultados de despliegue fluctuaban demasiado para cada una de las simulaciones sobre el mismo lote de prueba; mientras que, a partir de las 100000 corridas, los resultados se estabilizaron.

8.3. Resultados

8.3.1. Redes Estrella

Para las redes estrellas se realizaron 20 simulaciones, separando las mismas en dos grupos. Para el primer grupo se plantearon topologías Estrella puras y para el segundo se plantearon modelos Estrella con configuración jerárquica.

Como resultado de todo ello se observó que, para el primer tipo, la QoS de la topología se ve decrementada si se aumenta la cantidad de nodos Fog. Sin embargo, al incrementar la cantidad de nodos Fog, se incrementa momentáneamente la cantidad de despliegues, siempre dependiendo del enlace utilizado al Cloud y de la cantidad de aplicaciones desplegadas, como así también de los dispositivos a los cuales requieren acceso las aplicaciones. A su vez, si se incrementa la cantidad de módulos de cómputo (IoTController) a desplegar en la topología, aumenta la cantidad de despliegues y la QoS no se ve disminuida respecto del Cloud.

Los valores de Calidad de Servicio en este tipo de red rondan entre el 50% y el 80%, dependiendo de la cantidad de fogs y del tipo y cantidad de aplicaciones a desplegar, entregando los mejores valores cuando se considera un solo nodo de red conectando ambos extremos.

En cuanto a la utilización de recursos Fog Computing, en topologías con un solo nodo Fog, la misma ha llegado a ser del 84%, mientras que el promedio del resto de los despliegues -incluso en ambos tipos de red Estrella- se mantiene sobre el 30%.

Aquellos despliegues en donde se aloja una o más de una aplicación en un Fog, la consumición de recursos aumenta, sin depender de la cantidad de dispositivos a los que requiere acceso la/s misma/s ni del tamaño de sus máquinas virtuales. La Figura 2 muestra los resultados de una red de este tipo.

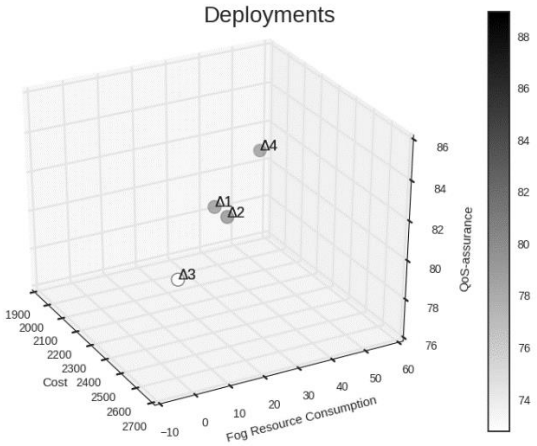


Figura 2: Red Estrella Pura de 20 dispositivos. Δ4: [IoTController->fog1][Dashboard->fog1][DataStorage->cloud1]

Respecto al segundo grupo de redes Estrella se obtuvieron muy pocos despliegues, de los cuales se puede observar una tendencia a utilizar solo el Cloud. La jerarquía presenta una alta sensibilidad a los tipos de enlaces empleados y a la cantidad de nodos Fog que conforman la misma. La Calidad de Servicio de los despliegues entregados es superior al 50%, pero la mayoría de las pruebas arrojaron despliegues en los

que el Cloud aloja la mayor cantidad de módulos software.

8.3.2. Redes Peer-to-Peer

Para este tipo de topología se planteó un solo grupo de evaluación, el cual fue mutando en cuanto a la cantidad de conexiones de pares existentes, lo que condujo a una evolución durante la formulación de los lotes de prueba. Siempre considerando una red de 4 fogs, se comenzó conectando algunos de ellos, para luego incrementar los enlaces en base a los resultados que se iban observando y terminar constituyendo una red de pares como la teoría la define (todos con todos).

Es por ello que los resultados de las simulaciones topológicas de Peer-to-Peer arrojaron como conclusión más importante que, contra más pares (nodos Fog) existan conectados, mayor es la cantidad de despliegues posibles, y, si se incrementa a su vez la cantidad de módulos de cómputo que tienen acceso a los dispositivos finales, mayor aún es la cantidad de despliegues. Sin embargo, la consumición de recursos Fog, independientemente del planteo topológico, nunca supera el 30%. La Figura 3 muestra una red de pares incompleta, es decir, sus nodos no están totalmente conectados, pero ya se observa la tendencia en el aumento de la cantidad de despliegues y su reacción en cuanto a QoS.

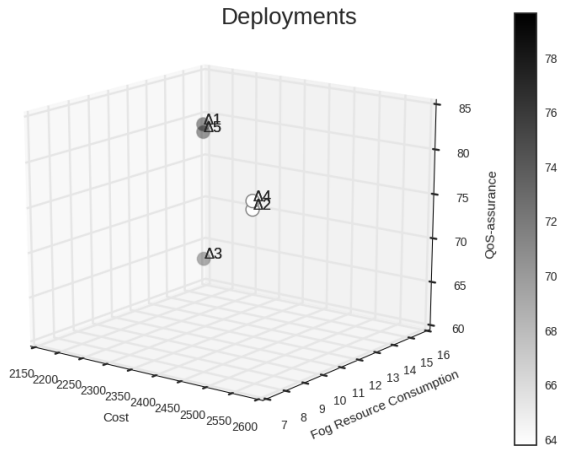


Figura 3: Red Peer-to-Peer de 60 dispositivos. Δ1: [IoTController->fog3][Dashboard->fog3][DataStorage->cloud1]

Si bien todo indica que la ubicación de las aplicaciones o módulos más pesados se realice en los nodos Fog más potentes o capaces, la dispersión de estos en los diferentes fogs no contribuye de buena manera a la QoS de la red en general.

En números, la Calidad de Servicio de la red en este tipo de topología no supera el 80%, aunque se plantearon pruebas con variabilidad de links desde los nodos Fog hacia el Cloud, colocando un enlace de mayor latencia y menor ancho de banda, y se obtuvieron resultados con QoS superiores al 85%, con consumiciones de recursos Fog del 25% en promedio y, claramente, una tendencia a la utilización mayoritaria del Cloud, cargando sobre los nodos de la red solo aquellas aplicaciones más livianas.

8.3.3. Redes Mesh

En el último tipo de red se consideraron dos grupos diferenciados por su estructura: por un lado, una red en la que existe un nodo Fog aislado conectando al extremo inferior con el Cloud y 3 fogs conectados entre sí con una sola salida al Cloud, y por el otro lado una red con 6 nodos Fog separando 3 en formato Peer-to-Peer y 3 en formato Estrella (Estrella pura y jerárquica).

En cuanto al primer grupo, los resultados arrojados conllevan a afirmar que la Calidad de Servicio de la Red es menor que las redes Estrella, a excepción de cuando se centraliza la comunicación con el Cloud. Normalmente, cuando la red contiene una parte estrella que se comunica directamente con el Cloud la QoS es superior al 60%. Ejemplo de ello es la Figura 4.

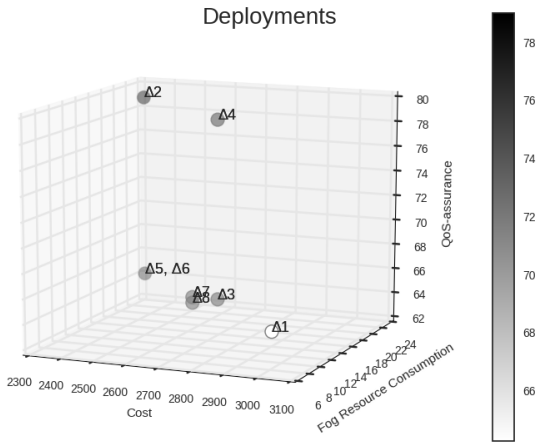


Figura 4: Red Mesh con sector estrella y Peer-to-Peer de 60 dispositivos. Δ2: [IoTControllerA->fog4][IoTControllerB->fog4][Dashboard->fog4][DataStorage->cloud1]

En la mayoría de los casos, el despliegue de las aplicaciones se da en los nodos raíz de las secciones estrella y en los nodos más fuertes de las secciones Peer-to-Peer, arrojando como consumición de recursos un promedio no superior al 25%.

Por el otro lado, en el segundo grupo, se observaron patrones de comportamiento repetitivos a los arrojados por todas las pruebas realizadas anteriormente, la QoS de las redes decrementa cuando existen una mayor cantidad de links entre los nodos de red, otorgando garantía de Calidad de Servicio de no más del 45%. Al igual que en el grupo

anterior, la consumición de recursos tampoco supera el 30%.

8.4. Resultados Generales

Se observaron ciertas particularidades durante el proceso de simulación que son descriptas a continuación:

Respecto a la Garantía de Calidad de Servicio de los despliegues entregados, la misma se ve alterada cuando el tipo de enlace varía la latencia y el ancho de banda, de modo que los enlaces de menor ancho de banda y mayor latencia (Satelital) siempre entregaron un mayor porcentaje de garantía, en aquellos casos donde existieron despliegues posibles.

La cantidad de despliegues también varía dependiendo de la conectividad de los dispositivos de red, como así también por el tipo de enlace utilizado para comunicarse con el Cloud, de modo que, a mayor latencia y menor ancho de banda, existe una reducida cantidad de despliegues posibles.

En cuanto al porcentaje de consumición de recursos hardware en los nodos de red producido por la carga de aplicaciones sobre estos, presenta una clara influencia el tamaño de las máquinas virtuales o recursos hardware requeridos por las aplicaciones a desplegar sobre los mismos.

En cuanto a la variable de costos, en aquellas topologías donde la utilización de recursos Fog es mayor, el costo de utilización de las mismas es inferior comparada a aquellas en que se utilizan pocos recursos hardware de los nodos de red, requiriendo más del procesamiento y almacenamiento del Cloud.

9. Conclusiones

Luego de embeber las diferentes topologías de red más utilizadas para IoT en el modelo Fog Computing y realizar las simulaciones correspondientes sobre el prototipo Fog Torch Pi, se ha llegado a la conclusión de que aquellas redes Estrella puras, con centralización de procesamiento y que realizan la comunicación de los extremos de la red por medio de un nodo Fog, cumplen con una garantía de Calidad de Servicio -arrojada por el simulador empleado- superior a la del resto de las topologías (Peer-to-Peer y Mesh), alcanzando valores del 80% de satisfacción de la QoS demandada.

Se toma en cuenta también que la disminución de costos que resulta de la utilización de este tipo de topología Estrella es de aproximadamente un 10% respecto de la utilización de solamente el Cloud y de un 5% aproximadamente respecto del resto de las topologías. Esto conduce a afirmar que, las redes Estrella, al menos en las situaciones planteadas, resultan ser más económicas que el resto de las redes planteadas.

A pesar de que el modelo de tipo Estrella presenta una mejor performance en cuanto a satisfacción de QoS respecto a los demás modelos, exhibe una menor cantidad de posibilidades de despliegue de aplicaciones en los nodos Fog. Las redes Peer-to-Peer son quienes facilitan ese despliegue de módulos de software de un ambiente Fog Computing a pesar de añadir redundancia e incrementar los riesgos de interferencia a través de sus links.

Por su parte, las redes Mesh, añaden a las ventajas características de las redes Estrella, la variedad de opciones de despliegue de las redes de pares, por lo que se entiende que, si se combinan correctamente las partes, las redes Mesh se vuelven más convenientes para determinadas circunstancias.

La escalabilidad en las redes ha demostrado en este estudio que, a mayor cantidad de dispositivos a controlar, mayor es la cantidad necesaria de nodos Fog a implantar en la red, exigiendo una capacidad suficientemente alta de los mismos para poder cargar las aplicaciones necesarias, lo que también se traduce en costos. Esto a su vez significa una mayor utilización de recursos Fog, sin garantizar demandas de QoS en un 100% se introduce una nueva discusión acerca de los servicios ofrecidos por el modelo Fog Computing en general.

Para finalizar, se puede concluir que es incondicional la presencia de un profesional en el tema que pueda evaluar y determinar las posibilidades y conveniencias en cada situación a emplear un modelo Fog. Este trabajo deja en claro que no existe aún un modelo estático excepcional para ser utilizado en todos los casos posibles y a gran escala. Se debe realizar un estudio previo de la situación para tomar dimensión principalmente del tamaño del problema o situación a resolver, para luego, considerando como base los resultados expuestos, poder sugerir un modelo conveniente y adaptable a la circunstancia.

10. Trabajos futuros

El grupo de trabajo continúa simulando patrones de configuración que puedan indicar la conveniencia en la utilización del modelo Fog Computing en un modelo topológico estático en ambientes tan heterogéneos como los de IoT. Para ello, se está utilizando un simulador que, por defecto, permite la conformación y evaluación de topologías a gran escala. Dicha herramienta se encuentra en fase de

estudio y se espera obtener resultados que contrasten los ya obtenidos.

Referencias

[1] P. Mell, T. Grance, "The NIST definition of cloud computing", NIST Special Publication, 800 – 145, 2011.

[2] CBS Interactive Inc, "Special Report: The future of Everything as a Service", 2017.

[3] J. Biron, J. Follett, "Foundational Elements of an IoT Solution The Edge, The Cloud, and Application Development", O'Reilly Media, Inc., 2016.

[4] Garnet, "Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016, Up 30 Percent From 2015", http://www.gartner.com/newsroom/id/3165317, 2016.

[5] B. Varghese, N. Wang., D.Nikolopoulos. R.Buyya, "Feasibility of Fog Computing", arXiv:1701.05451v1cs.DC, 2017.

[6] Y. Ai, M.; Peng, K. Zhang, "Edge cloud computing technologies for internet of things: A primer", Digit. Commun, Netw. 2017, in press.

[7] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for Internet of Things and analytics", Big Data and Internet of Things: A Roadmap for Smart Environments, Cham, Switzerland: Springer, pp. 169–186. 2014.

[8] G. Premsankar, M. Di Francesco, T.Taleb, "Edge Computing for the Internet of Things: A Case Study", IEEE INTERNET OF THINGS JOURNAL, VOL. 5, NO. 2, APRIL 2018.

[9] H. F. Atlam, R. J.Walters, G. B. Wills, "Fog Computing and the Internet of Things: A Review", Big Data Cogn. Comput, 2018, 2, 10; doi:10.3390/bdcc2020010.

[10] M. Chiang, T. Zhang, "Fog and IoT: An Overview of Research Opportunities", IEEE Internet Things J. 2016, 3,854–864.

[11] J. Dilley et al., "Globally Distributed Content Delivery". IEEE Internet Computing, vol. 6, No.5, 2002, pp.50–58, 2002.

[12] M. Satyanarayanan, "Pervasive Computing: Vision and Challenges", IEEE Personal Comm., vol. 8 No.4, 2001, pp.10–17.

[13]M.Satyanarayanan, P. Bahl, R. Cáceres, N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing", PERVASIVE computing, 2009 IEEE.

[14] F. Bonomi et al., "Fog Computing and Its Role in the Internet of Things", Proc. 1st Edition MCC Workshop Mobile Cloud Computing (MCC12), 2012, pp.13–15.

[15] C. Hewitt, P. Bishop, R. Steiger, "A Universal Modular ACTOR Formalism for Artificial Intelligence", Proceedings of the 3rd international joint conference on Artificial, 1973.

[16] A. V. Dastjerdi, R. Buyya, "Fog Computing: Helping the Internet of Things Realize Its Potential", Computer, IEEE Computer Society - 0018-91 62/16 /2016.

[17] N. Rodríguez, M. Murazzo et al.: "Estudio de las mejoras de aplicar Fog Computing en la distribución de servicios en Cloud Computing", XIX WICC. Bs Aires, 2017.

[18] Jekishan K. Parmar, Ankit Desai. "IoT: Networking Technologies and Research Challenges", International Journal of Computer Applications, Volume 154, November 2016.

[19] Antonio Brogi, Stefano Forti, Ahmad Ibrahim, "How to best deploy your Fog applications, probably", 1st IEEE International Conference on Fog and Edge Computing (ICFEC 2017), May 14th 2017, Madrid, Spain, http://pages.di.unipi.it/forti/pdf/icfec17.pdf.

[20] Antonio Brogi, Stefano Forti, Ahmad Ibrahim, "Predictive Analysis to Support Fog Application Deployment", Preprint, 2018.

[21] CISCO, "Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are", 2015. White Paper: www.cisco.com/go/iot.

[22] Sillero Cristian J., "Fog Computing y topologías IoT, más convenientes que el Cloud Centralizado", 5º CONAIISI, Santa Fé, 2017.

Implementing cloud-based parallel metaheuristics: an overview

Patricia González[1], Xoán C. Pardo[1], Ramón Doallo[1], and Julio R. Banga [2]

[1]*Computer Architecture Group, Universidade da Coruña, Spain*
{patricia.gonzalez, xoan.pardo, doallo}@udc.es
[2]*BioProcess Engineering Group, IIM-CSIC, Spain*
julio@iim.csic.es

Abstract

Metaheuristics are among the most popular methods for solving hard global optimization problems in many areas of science and engineering. Their parallel implementation applying HPC techniques is a common approach for efficiently using available resources to reduce the time needed to get a good enough solution to hard-to-solve problems. Paradigms like MPI or OMP are the usual choice when executing them in clusters or supercomputers. Moreover, the pervasive presence of cloud computing and the emergence of programming models like MapReduce or Spark have given rise to an increasing interest in porting HPC workloads to the cloud, as is the case with parallel metaheuristics. In this paper we give an overview of our experience with different alternatives for porting parallel metaheuristics to the cloud, providing some useful insights to the interested reader that we have acquired through extensive experimentation.

Keywords: parallel metaheuristics, cloud computing, MPI, MapReduce, Spark

1 Introduction

Optimization is concerned with finding the ”best available” solution for a given problem. Many key problems in different areas of science, economics and engineering can be formulated and solved using different optimization techniques [1, 2, 3]. For example, optimization problems are playing an increasing role in computational biology, bioinformatics and chemistry, helping in the development of novel therapies and drugs for different diseases such as cancer or autoimmune diseases. Most of these optimization problems are, in practice, NP-hard, complex, and time-consuming. Stochastic global optimization methods are robust alternatives to solve these problems. And among them, metaheuristics are gaining increased attention as an efficient way of solving hard global optimization problems. However, in most realistic applications, metaheuristics still require large computation time to obtain an acceptable result.

Thus, the parallelization of optimization methods in general, and of metaheuristics in particular, and the use of HPC resources, like clusters or supercomputers, have been a common approach to speed-up the computations, increase the size of the problems that can be handled or attempt a more thorough exploration of the solution space. Furthermore, the technological developments of the last decades, continuously reducing the cost/performance ratio of HPC resources, have made accessing to them more feasible. However, when considering very challenging problems, the provision of a large number of resources, which is not always practicable, is essential for the success of the parallel solution. The emergence in the last years of cloud computing [4] as a new model for the effortless provision of a large number of computing resources has attracted the attention of the HPC community.

Cloud computing has some specific characteristics that make it an interesting alternative for the provision and management of computing resources. Public cloud providers offer self-service interfaces to external users for the on-demand provision of virtualized computing resources. Users share the provider infrastructure while having exclusive zero-queue-time access to their own isolated virtual resources, configured to their applications' needs. Resource consumption is charged using a pay-as-you-go model at very appealing prices. Additionally, collaborators can be provided with controlled access to resources and environments be shared with third-parties using virtual machine images.

But despite these appealing features, its adoption by the HPC community has been limited. The main reason is the performance obtained when using traditional parallel programming models, like MPI, on virtual clusters. Several evaluations [5, 6] have shown that although computationally-intensive applications present little overhead, communication-intensive ones

have poor performance thus limiting their scalability.

On the other hand, to facilitate the development of large-scale distributed applications, new programming models like MapReduce [7] or Spark [8] have been proposed. These models combine high-level programming abstractions and distributed execution frameworks with implicit support for deployment, data distribution, parallel processing and run-time features like fault tolerance or load balancing. But, although their many advantages, these proposals also have some shortcomings that could discourage their use for HPC workloads. They are designed with the analytics of big amounts of data in mind, preferring availability to performance and providing lower speedup and distributed efficiency than traditional parallel frameworks.

In this paper we give an overview of our experience with the different alternatives we have studied for porting some representative parallel metaheuristics to the cloud. By means of extensive experimentation, using both synthetic and real-world benchmarks on different traditional and cloud testbeds, we have analyzed the pros and cons of the different options. Considering the demanding and dynamic nature of the implementation of these methods, we think that they are exemplary of the many applications that could benefit from being ported to the cloud, and that the conclusions drawn could be useful in many other scenarios.

The rest of the paper is structured as follows. Section 2 describes the alternatives for writing parallel programs for the cloud that we have analyzed. Section 3 introduces the Differential Evolution metaheuristic and the parallel versions we have implemented. In section 4 the main lessons learned from experimental results are overviewed and justified. Section 5 references some related work. Finally, Section 6 summarizes the conclusions of the paper.

2 Parallel programming in the cloud

Nowadays there are different alternatives that could be used to implement parallel programs for the cloud. Examples include "traditional" message-passing or many-task computing approaches, using actors or using data-oriented models and abstractions like MapReduce (MR) or Resilient Distributed Datasets (RDD). From these alternatives, we have evaluated and compared three different approaches. Each of them is based on different programming abstractions, implemented using different programming languages and executed using different platforms:

1. A C implementation using MPI in a virtual cluster.

2. A Java implementation using MR in a Hadoop cluster.

3. A Scala implementation using the RDD abstraction in a Spark cluster.

The **Message Passing Interface (MPI)** is the de-facto standard for HPC distributed computing. It is a standard interface that allows developers to write parallel applications in C, C++ or FORTRAN using the message-passing model. In this model, parallel programs consist of a set of processes that communicate with each other by sending and receiving messages. Processes have separate address spaces and the programmer has explicit control over the memory used by each process and how the communication occurs.

MapReduce (MR) [7] is a programming model and associated distributed execution framework originally proposed for processing large datasets in commodity clusters. For many years it has been the de-facto standard for cloud programming. A program in MR is composed of a pair of user-provided map and reduce functions, generally written in Java, that are executed in parallel over a distributed network of worker processes. Executions can be described as a directed acyclic data flow where a network of stateless mappers and reducers process data in single batches, using a distributed filesystem (typically HDFS) to take the input and store the output. The execution framework has a master-worker architecture with implicit support for data distribution, parallel processing and fault tolerance.

Resilient Distributed Datasets (RDD) [8] is the main abstraction provided by Spark for supporting fault-tolerant and efficient distributed in-memory computations. An RDD is a read-only fault-tolerant partitioned collection of records that is created from other RDDs or from data in stable storage by means of coarse-grained transformations (e.g., map, filter or join) that apply the same operation to many data items at once. RDDs are used in actions (e.g. count, collect or save) to return a value to the application or export data to a storage system. Spark provides a programming interface to RDDs for Scala, Java or Python, and a distributed execution framework composed of a single driver program and multiple workers, that are long-lived processes that persist RDD partitions across operations. Developers write the driver program in which they can manipulate RDDs using a rich set of operators, control their partitioning to optimize data placement and explicitly persist intermediate results to memory or disk.

Each of the selected approaches has its pros and cons. The MPI approach is HPC oriented and its main advantage is the high communications performance combined with the use of a compiled programming language. The other two approaches are HTC oriented, having JVM-based distributed execution frameworks and using interpreted or JVM languages, which a priori will provide worst performance. On the other hand, compared with the other two, the MPI approach is too low level, requiring that the programmer deals with details like the data distribution, the ordering of communications or the fault-tolerance support.

Both MR and Spark provide high-level data abstractions, interfaces to object and/or functional languages and distributed execution frameworks with implicit support for data distribution and fault tolerance, so programming is easier and less error-prone. The main difference between them is how they support iterative algorithms. MR has been designed to execute programs in batches, taking input, and storing output and intermediate data in the file system. Executing iterative algorithms has considerable overhead because there is no way of efficiently reusing data or computation from previous batches. On the contrary, Spark has been designed with iterative algorithms in mind, providing efficient in-memory processing between iterations.

3 Parallel implementation of the Differential Evolution metaheuristic

For our study we have selected one representative population-based metaheuristic that will be used as a test case, the Differential Evolution (DE) [9]. DE is very popular and has been successfully applied to many problems [10]. It is an stochastic iterative method (see algorithm 1) that starting from an initial population matrix composed of NP D-dimensional solution vectors (individuals), tries to achieve the optimal solution iteratively using vector differences to create new candidate solutions. In each iteration, new individuals are generated by means of operations (crossover - CR; mutation - F) performed among individuals in the population matrix. New solutions will replace old ones when its fitness value was better. A population matrix with optimized individuals is obtained as output of the algorithm. The best of which is selected as the solution close to optimal for the objective function of the model.

Different models have been proposed for the parallelization of metaheuristics [11]. We have considered two of the most popular:

1. The *master-slave* model, that preserves the behaviour of the sequential algorithm parallelizing the inner-loop of the DE. In this model a master processor distributes computation operations between the slave processors.

2. The *island-based* model, that divides the population in subpopulations (*islands*) where the DE algorithm is executed isolated. To preserve diversity and avoid getting stuck in local optima sporadic individual exchanges are performed among islands.

Our first approach was to parallelize the DE using the master-slave model. But when we were implementing it with Spark, we found an issue with the parallelization of the generation of new individuals. In the implementation of this model with Spark, the population is partitioned and distributed among workers.

Algorithm 1 Differential Evolution algorithm

Input: A population matrix P of size D x NP
Output: A matrix P whose individuals were optimized

repeat
  for each element i of the P matrix **do**
    choose randomly different r_1, r_2, $r_3 \in [1, NP]$
    choose randomly an integer $j_r \in [1, D]$
    for $j \leftarrow 1, D$ **do**
      choose a randomly real $r \in [0, 1]$
      if $r \leq CR$ or $j = j_r$ **then**
        $u_i^{G+1}(j) \leftarrow x_{r1}^G(j) + F \cdot (x_{r2}^G(j) - x_{r3}^G(j))$
      else
        $u_i^{G+1}(j) \leftarrow x_i^G(j)$
      end if
    end for
    evaluate (u_i^{G+1})
    if $fitness(u_i^{G+1}) < fitness(x_i^G)$ **then**
      $x_i^{G+1} \leftarrow u_i^{G+1}$
    else
      $x_i^{G+1} \leftarrow x_i^G$
    end if
  end for
until Stop conditions

For the mutation of each individual, three random different individuals have to be selected from the whole population but, unlike MPI, communication among workers is not allowed in Spark. Each worker only have access to individuals of its partition and the driver is the only that has access to the complete population.

From the alternatives we tried to deal with this problem, broadcasting a read-only copy of the whole population to every worker in each iteration was the option that has shown best benchmarking results. But even so, the communications overhead introduced by this solution was unfeasible. So we concluded that the master-slave model did not fit well with the distributed programming model of Spark and decided to discard it in favour of the island-based model. A complete discussion of the implementation of the master-slave model and the other alternatives we tried to deal with this issue can be found in [12].

3.1 Island-based DE implementations

Dividing the population into subpopulations reduces interprocessor communications and leads to more loosely coupled parallel applications. Therefore, at least theoretically, the island model should be more suitable for implementing parallel metaheuristics with programming abstractions like MR or RDDs. In this section we explain the basics of the three island-based implementations of the DE that we have studied and compared.

Figure 1 shows the schematic diagram of the implementation with MPI of an asynchronous island-based DE (`asynPDE`) first presented in [13]. Each process ex-

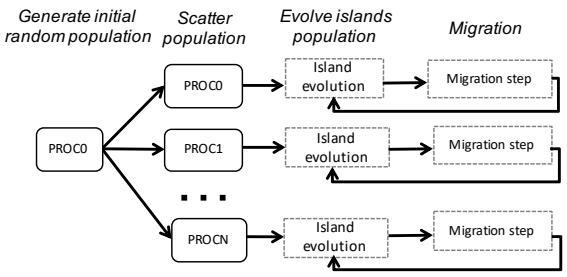


Figure 1: Island-based DE implementation with MPI

ecutes the same inner-loop of the sequential DE, doing mutation and crossover operations within the subpopulation of its island. A migration step is performed once every few iterations to migrate individuals among islands in order to preserve diversity. MPI asynchronous communications have been used to avoid having idle processes waiting for migrants arriving from other processes. The arriving of new individuals is examined at the end of the migration step (algorithm 2), proceeding with the next island evolution if new solutions have not yet arrived. Missed migrations will be checked again later, after each evolution of the island (algorithm 3). Asynchronous communications are also used for checking the stopping criteria of the algorithm. Whenever a process reaches a stopping condition, it sends a message to the rest so they can all stop almost at the same time.

As every MR program, our implementation of the island-based DE with MR (`mrPDE`) presented in [14] has two components: the main program (*driver*) that will be executed by the master and the *map* and *reduce* functions that will be executed by the workers. In the driver (algorithm 4) the outer-loop of the DE is executed to evolve a randomly-generated population until the stopping criterion is met. In every iteration, the population is partitioned into islands that are written to HDFS, a batch MR job is launched to evolve the islands in parallel and the resulting global population is read from HDFS once the job finished. To introduce diversity individuals are randomly shuffled during the partition of the population. The inner-loop of the DE algorithm is executed in the map function (algorithm 5) of the MR job. Each map instance reads an island population from HDFS and applies the mutation strategy during a predefined number of evolutions, taking random individuals from its island only. Finally, an output record is emitted for each individual of the evolved island using its fitness value as key. The MR job is completed with a single identity reducer that writes all the population to HDFS ordered by fitness.

The schematic diagram of our island-based DE implementation with Spark (`SiPDE`) presented in [12] is shown in figure 2. The population has been represented as a key-value pair RDD (solid outlined boxes in the figure). Each partition of the population RDD is considered to be an island (shaded rectangles in the figure, darker if they are persisted in memory). The algorithm starts generating in parallel an initial random population using a *map* transformation. Then, an *evolution-migration* loop is repeated until the stopping criterion, implemented as a *reduce* action (a distributed

Algorithm 2 Migration step of the MPI impl.

```
1: select(migSet);
2: // asynchronous send
3: ISend(migSet, remoteDest);
4: // asynchronous reception (non-blocking)
5: IRecv(recSet, remoteDest);
6: Test(recSet, isComplete);
7: if isComplete then
8:     Replace(recSet);
9: end if
```

Algorithm 3 Island evolution of the MPI impl.

```
1: repeat
2:     for each element in the population do
3:         cross(); mutation(); eval();
4:     end for
5:     while pending migration do
6:         // check pending messages (non-blocking)
7:         Test(recSet, isComplete);
8:         if isComplete then
9:             Replace(recSet);
10:        else
11:            break;
12:        end if
13:    end while
14: until Stop conditions
```

Algorithm 4 Driver of the MR implementation

Input: DE configuration parameters
Output: A population P of optimized individuals

```
1: P ← initial random population
2: #i ← number of islands
3: repeat
4:     // partition and shuffle the population
5:     Islands⃗ ⇐ PartitionPopulation(P, #i)
6:     P ⇐ EvolveIslands(Islands⃗) // the MR job
7: until Stop conditions
```

Algorithm 5 Map function of the MR implementation

Input: An island I; DE configuration parameters
Output: An island I of optimized individuals

```
1: repeat
2:     // apply the DE mutation strategy
3:     I ⇐ EvolveIsland(I)
4: until number of evolutions
5: for each individual Ind⃗ of the island I do
6:     Emit(fitness(Ind⃗), Ind⃗)
7: end for
```

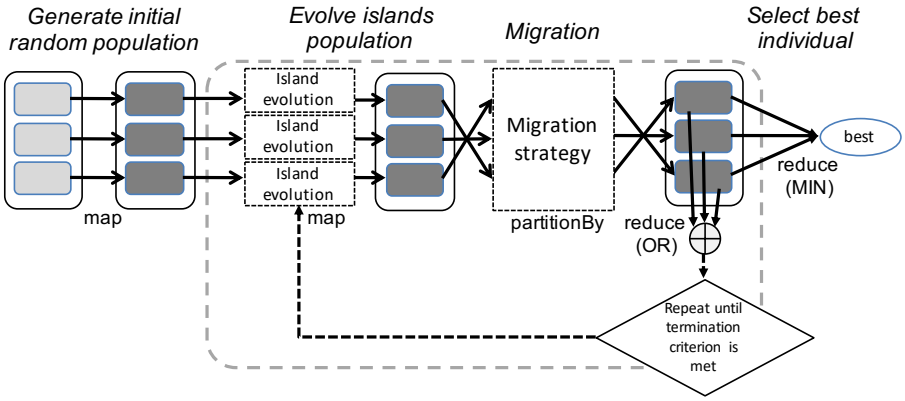


Figure 2: Island-based DE implementation with Spark

OR), is met. Finally, the best individual is selected using a *reduce* action (a distributed MIN). During each iteration of the *evolution-migration* loop islands evolve isolated for a predefined number of evolutions. After which, and to introduce diversity, the same random shuffling as in the MR implementation is performed using a custom random *partitioner*.

The three programming models have unique inherent features that have conditioned the implementations, modifying the systemic properties of the original algorithm, and resulting in different searches. As we will see later, this will influence the benchmarking results. The parts of the algorithm that have been implemented differently are:

- *The migration strategy*. While in MPI it is used an strategy with replacement (i.e. best individuals in one island replace worst individuals in the neighbour), in MR and Spark, as messaging between worker processes is not possible, the migration is a random shuffling of the population without replacement.

- *The evolution-migration synchronization*. In MPI individuals are migrated using asynchronous messaging between processes, so each island can proceed with a new evolution without waiting for the others. For the same reason as before, in MR and Spark that is not possible, so migrations introduce a synchronization step between island evolutions.

- *The stopping criterion checking*. In MPI whenever a process reaches a stopping condition sends an asynchronous message to the rest and all stop almost at the same time. Again that is not possible in MR and Spark, that have to wait until all the islands conclude their evolutions to check if any of them reached the stopping condition.

It must be noted that, although the migration steps in MR and Spark appear to be very similar, their implementations and overheads are very different. In MR the migration is performed in the *driver*, reading the population from HDFS, shuffling the individuals and writing back the islands to HDFS, while in Spark a custom random *partitioner* is used (no HDFS overhead).

4 Lessons learned from experimental results

In our previous works [14, 15, 16], we have conducted extensive testing of the three proposed island-based DE implementations using different benchmarks and execution platforms. In this section we overview the main lessons that we have learned, using the experimental results obtained for a challenging problem from the domain of computational systems biology, the *Circadian* model [17], as example. It is a parameter estimation problem in a nonlinear dynamic model of the circadian clock in the plant *Arabidopsis Thaliana*, that is known to be particularly hard due to its ill-conditioning and non-convexity.

Table 1 summarizes the results of the experiments with the MPI (`asynPDE`) and Spark (`SiPDE`) implementations. In the columns the number of cores (*#islands*) used, the mean number of evaluations required (*#evals*), the mean and deviation of the execution times (*time(s)*), and the speedup achieved versus the sequential execution are shown. Two different platforms have been tested varying the number of cores from 2 to 16/32 and calculating the mean of 20 independent runs for each configuration. The first platform was a cluster (named Pluton) with 16 nodes powered by two octa-core Intel Xeon E5-2660 @2.20GHz with 64GB and an InfiniBand FDR network. The second platform was a virtual cluster formed by A3 instances (4 cores, 7GB) in the Microsoft Azure cloud. The quality of the solution has been used as termination condition with a value-to-reach (VTR) of 1e-5 and, for the migration frequency, 50 evolutions between migrations have been used for `asynPDE` and 200 for `SiPDE`. Refer to [16] for the detailed experimental setting.

From table 1 we can conclude the following:

- *All the parallel implementations (including MR not shown in the table) have improved the conver-*

Table 1: MPI and Spark results (DE params: NP=256, D=13, CR=0.8, F=0.9, VTR=1e-5)

impl.	platform	#islands	#evals	time(s)	speedup
asynPDE	cluster	1	6,480,102	15,230.22±886.80	-
		2	3,540,889	4,078.36±1,852.32	3.73
		4	1,815,689	1,100.08±180.96	13.84
		8	1,231,094	380.99±77.64	39.97
		16	1,236,346	220.79±51.17	68.98
		32	1,700,782	149.82±30.37	101.65
	Azure	1	6,633,830	37,952.61±3,224.67	-
		2	3,067,622	9,196.63±1,110.82	4.13
		4	1,809,942	2,659.65±410.31	14.27
		8	1,279,609	929.77±204.21	40.82
		16	1,301,888	491.92±87.50	77.15
SiPDE	cluster	1	6,437,670	40883.39±3712.56	-
		2	5,980,416	19275.65±1281.63	2.12
		4	5,729,536	9305.30±909.41	4.39
		8	3,904,256	3319.33±296.88	12.32
		16	1,835,776	790.97±90.50	51.69
		32	1,577,216	348.36±43.47	117.36
	Azure	1	6,565,461	93,977.02±5216.28	-
		2	5,333,186	41,140.87±6474.26	2.28
		4	5,716,736	21,030.04±2443.06	4.47
		8	3,983,616	7,444.79±928.91	12.62
		16	1,953,536	1,768.25±166.51	53.15

gence time of the sequential algorithm. This was an expected result, given that, the evaluation of the population is performed in parallel. Moreover, superlinear speedups are obtained because, in the island model, the cooperation between islands improves the convergence rate.

- *The MPI implementation has the lower convergence rate (i.e. required number of evaluations), specially when using few cores*. This is due to the inherent features of each programming model explained in Section 3.1.

- *The convergence rate of the Spark implementation improves with the number of islands*, while the MPI implementation stagnates for more than 8. The reason is that the shuffling of the population, used as migration strategy in the Spark implementation, maintains the diversity when the number of islands increases.

- *The convergence rate has similar results in Azure, but execution time worsens and speedup improves*. The execution times are between 2x and 3x times worse in Azure than in the cluster due to the overhead of virtualization and using non-dedicated resources. By the contrary, the speedups are larger due to the computation-to-communication ratio, that is, the ratio between the time spent computing and communicating.

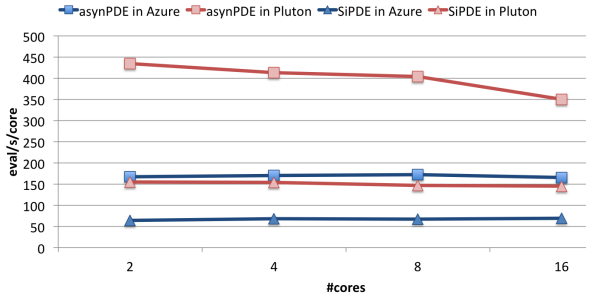


Figure 3: Evals/s/core of the MPI and Spark impl.

We have also calculated the number of evaluations per second and per core (*eval/s/core*) for both implementations and platforms (Figure 3). This is a good metric for evaluating the implementations because it includes not only the CPU time of evaluations, but also the communications and overhead time. From the figure we can conclude that:

- *The MPI implementation has a value of eval/s/core between 2.1x and 2.5x times better than the Spark implementation.*

- *The number of eval/s/core of the MPI implementation in the cluster drops with the growth of the number of cores*. This is because as the number of cores increases, so does the communications overhead. Besides, the computation of each is-

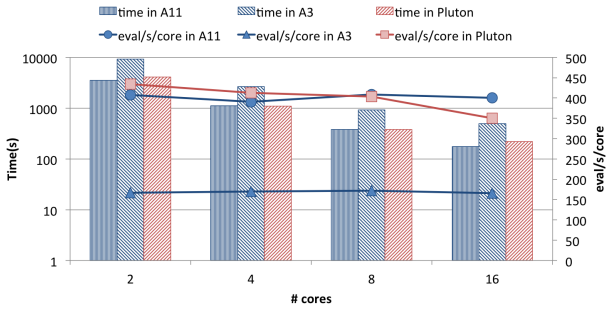


Figure 4: Comparison of MPI results in a physical cluster and in ordinary and HPC virtual clusters

land decreases, degrading the trade-off between computation and communication. Conversely, in the Spark implementation, the migration communication overhead is always the same, spreading the data movement among the available cores, and thus, having better scalability.

In order to evaluate the suitability of virtual clusters composed of HPC instances for running MPI applications, we have repeated the experiments with the MPI implementation in Azure, but now using a virtual cluster of A11 instances (16 cores, 112GB, Intel Xeon E5-2670 @2.6GHz). Figure 4 shows the results compared with those of table 1. From the figure we can conclude that:

- *The MPI implementation running in an HPC virtual cluster shows competitive execution times and better scalability.*

Figures 5 and 6 summarize the results of the experiments with the MR (`mrPDE`) and Spark (`SiPDE`) implementations in the cluster. Bean plots are used to show the execution times and the dispersion of the results. From the figures we can conclude that:

- *The MR implementation has larger execution times and dispersion than the Spark implementation.*
- *The MR implementation also reduces the convergence rate but with a limited speedup.*

To evaluate the overhead that is limiting the speedup of MR, we have measured, for a total of 8 iterations, the overhead of each *evolution-migration* iteration separately. These experiments were performed with versions of our implementations with the evolution of the population removed. Figure 7 shows the average and standard deviation of 10 independent runs of each experiment in two different platforms: the cluster we used before, and a virtual cluster formed by `m3.medium` instances (1 core, 3.75GB) in the AWS cloud. Refer to [14] for the detailed experimental setting. From the figure we can conclude that:

- *MR has significant higher overhead and larger dispersion than Spark.* It must be noted that,

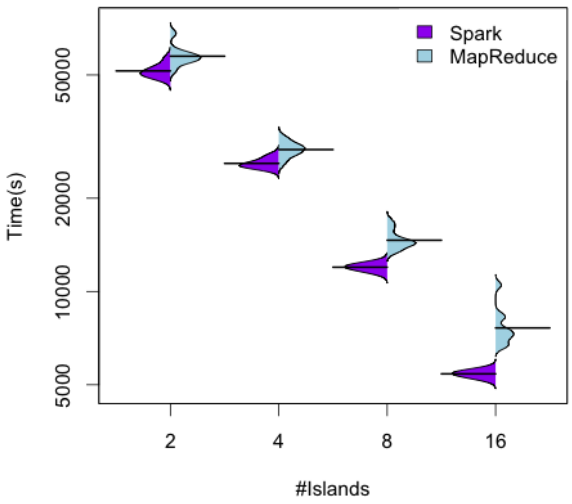


Figure 5: MR and Spark results in the cluster (DE params: NP=640, D=13, CR=0.8, F=0.9, VTR=1e-5)

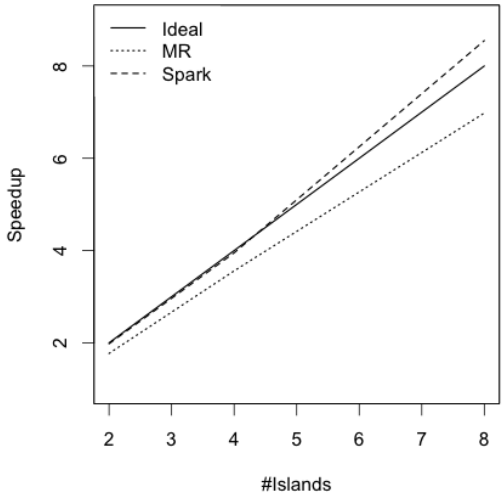


Figure 6: MR and Spark speedup in the cluster

although at first this would advise against using it in favour of Spark, for problems where the computation time significantly dominates over the overhead introduced by the iterations, MR would be competitive with Spark though with worst scalability.

- *Spark has better support for iterative algorithms than MR.* This was an expected result, given that, Spark persists results in-memory between iterations. That explains why the first iteration (the outliers in the box plots) is the most time consuming in Spark, while there is no significant difference between iterations in MR.
- *The overhead of MR and Spark is larger in AWS than in the cluster.* Spark is between 4x and 5x times worst in AWS, and MR around 2,8x. Also it must be noted that, in tune with what was expected, the Spark overhead slightly increases

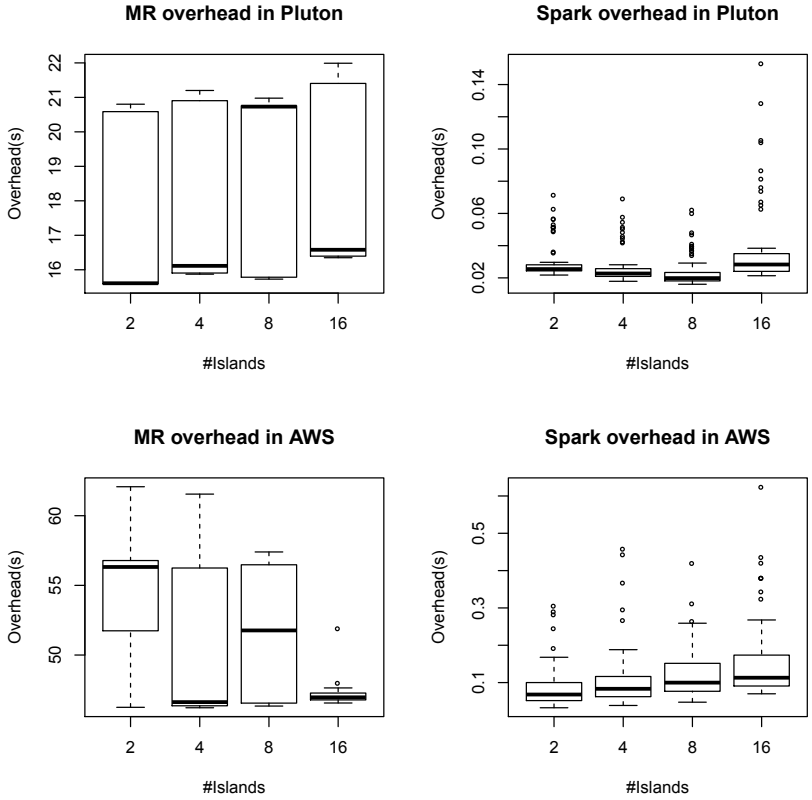


Figure 7: MR and Spark overhead per evolution-migration iteration

when the number of nodes grows.

5 Related work

Many works have been published analyzing the performance of HPC applications in the cloud. In [5, 6] extensive analysis to detect the more critical issues and bottlenecks are presented. These works conclude that the lack of high-bandwidth low-latency networks and the virtualization overhead has a great impact on the performance of tightly-coupled HPC workloads, as is the case of MPI applications.

There are also many works on the parallelization of metaheuristics. Nice reviews using traditional HPC approaches can be found in [11, 18]. With regard to the parallelization of DE, a distributed implementation using an island-model with asynchronous communications is proposed in [19].

Some works have studied the use of cloud programming models for the parallelization of metaheuristics. Most of these references are based on the use of MR, since not so long ago it was the de-facto standard for cloud programming. MRPSO, a parallelization of the Particle Swarm Optimization (PSO) metaheuristic is proposed in [20]. Different approaches to the implementation of parallel Genetic Algorithms (GA) are proposed in [21, 22]. In [23] different algorithmic patterns of distributed Simulated Annealing (SA) are designed and evaluated on Azure. In [24], a practical framework to infer large gene networks using a parallel hybrid GA-PSO is proposed.

There are also some references proposing parallelizations of the DE algorithm with MR. In [25] the fitness evaluation is performed in parallel, but experimental results show that the HDFS overhead reduces the benefits of the parallelization. In [26], a concurrent implementation of the DE steady-state model is proposed, but its applicability is limited to shared-memory architectures. In [27] it is proposed a parallel implementation of a DE-based clustering algorithm.

As Spark replaces MR as the new de-facto standard for cloud programming, the number of references using it is gradually increasing. A PSO proposal for data clustering in learning analytics can be found in [28]. PSO is also used in [29] to test the proposal of a parallel metaheuristic data clustering framework. In [30] a GA is parallelized for pairwise test generation. A MAX-MIN Ant System algorithm (MMAS) is presented in [31] to solve the Traveling Salesman Problem (TSP). A coral reef (CR) algorithm is applied to the job shop scheduling problem (JSP) in [32]. A Binary Cuckoo Search algorithm is applied to the crew scheduling problem (CrSP) in [33].

The first parallelization of the DE algorithm using Spark was proposed in [34]. However, only the fitness evaluation is performed in parallel following a

master-slave approach. An entire parallelization of the algorithm was first explored in [12]. In that paper we propose and evaluate implementations of the master-slave and the island-based models. Results showed that the island-based model is by far the best suited to be implemented using Spark. A thorough evaluation of the island-based implementation can be found in [15] and extensive comparatives of the implementations overviewed in this paper in [14, 16].

6 Conclusions

In this paper we overview the insights we have learned, through extensive experimentation, about some approaches to implement parallel metaheuristics in the cloud. Using the Differential Evolution metaheuristic and the Circadian model, a difficult parameter estimation problem from computational systems biology, as test case, three different programming paradigms: MPI, MapReduce and RDDs, have been evaluated on three different platforms: a cluster, the Azure cloud and the AWS cloud. Results show that, as it was expected, MPI outperforms the other approaches in terms of execution time, due to its reduced overhead and low level programming interface. Nevertheless, Spark should be positively consider when looking for better scalability, easier programmability and implicit support for data distribution and fault-tolerance. In our experience, to get efficient implementations using cloud programming models it is necessary to reshape the existing algorithms or to propose new ones.

The source code of the MPI and Spark implementations are publicly available at `https://bitbucket.org/xcpardo/sipde` and `https://bitbucket.org/pglez/asynpde`, respectively.

Acknowledgements

This research received financial support from the Spanish Government through the projects DPI2017-82896-C2-2-R, and TIN2016-75845-P (AEI/FEDER, UE), and from the Galician Government under the Consolidation Program of Competitive Research Units (Network Ref. R2016/045 and Project Ref. ED431C 2017/04), all of them co-funded by FEDER funds of the EU. We also acknowledge Microsoft Research for being awarded with a sponsored Azure account.

Competing interests

The authors have declared that no competing interests exist.

References

[1] C. A. Floudas and P. M. Pardalos, *Optimization in computational chemistry and molecular biology: local and global approaches*, vol. 40. Springer Science & Business Media, 2013.

[2] J. R. Banga, "Optimization in computational systems biology," *BMC systems biology*, vol. 2, no. 1, p. 47, 2008.

[3] I. E. Grossmann, *Global optimization in engineering design*, vol. 9. Springer Science & Business Media, 2013.

[4] I. Foster and D. B. Gannon, *Cloud Computing for Science And Engineering*. The MIT Press, 2017.

[5] R. R. Expósito, G. L. Taboada, S. Ramos, J. Touriño, and R. Doallo, "Performance analysis of HPC applications in the cloud," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 218–229, 2013.

[6] I. Sadooghi, J. H. Martin, T. Li, K. Brandstatter, K. Maheshwari, T. P. P. de Lacerda Ruivo, G. Garzoglio, S. Timm, Y. Zhao, and I. Raicu, "Understanding the performance and potential of cloud computing for scientific applications," *IEEE Transactions on Cloud Computing*, vol. 5, no. 2, pp. 358–371, 2017.

[7] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," in *The 6th USENIX Symposium on Operating Systems Design and Implementation*, 2004.

[8] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2012.

[9] K. V. Price, R. M. Storn, and J. Lampinen, *Differential Evolution - a practical approach to global optimization*, vol. 141. 01 2005.

[10] S. Das and P. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[11] E. Alba, G. Luque, and S. Nesmachnow, "Parallel metaheuristics: recent advances and new trends," *International Transactions in Operational Research*, vol. 20, no. 1, pp. 1–48, 2013.

[12] D. Teijeiro, X. C. Pardo, P. González, J. R. Banga, and R. Doallo, *Applications of Evolutionary Computation: 19th European Conference, EvoApplications 2016, Proceedings, Part II*, ch. Implementing Parallel Differential Evolution on Spark, pp. 75–90. 2016.

[13] D. R. Penas, J. R. Banga, P. González, and R. Doallo, "Enhanced parallel differential evolution algorithm for problems in computational systems biology," *Applied Soft Computing*, vol. 33, pp. 86 – 99, 2015.

[14] D. Teijeiro, X. C. Pardo, D. R. Penas, P. González, J. R. Banga, and R. Doallo, "Evaluation of parallel differential evolution implementations on mapreduce and spark," in *Euro-Par 2016: Parallel Processing Workshops*, pp. 397–408, Springer International Publishing, 2017.

[15] D. Teijeiro, X. C. Pardo, P. González, J. R. Banga, and R. Doallo, "Towards cloud-based parallel metaheuristics: A case study in computational biology with differential evolution and spark," *The International Journal of High Performance Computing Applications*, 2016.

[16] P. González, X. C. Pardo, D. R. Penas, D. Teijeiro, J. R. Banga, and R. Doallo, "Using the cloud for parameter estimation problems: Comparing spark vs mpi with a case-study," in *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid '17)*, pp. 197–206, 2017.

[17] J. Locke, A. Millar, and M. Turner, "Modelling genetic networks with noisy and varied experimental data: the circadian clock in arabidopsis thaliana," *Journal of Theoretical Biology*, vol. 234, no. 3, pp. 383–393, 2005.

[18] T. G. Crainic, "Parallel meta-heuristics and cooperative search," *CIRRELT*, vol. 58, 2017.

[19] J. Apolloni, J. Garca-Nieto, E. Alba, and G. Leguizamn, "Empirical evaluation of distributed differential evolution on standard benchmarks," *Applied Mathematics and Computation*, vol. 236, pp. 351–366, 2014.

[20] A. W. McNabb, C. K. Monson, and K. D. Seppi, "Parallel PSO using MapReduce," in *IEEE Congress on Evolutionary Computation, CEC2007*, pp. 7–14, IEEE, 2007.

[21] C. Jin, C. Vecchiola, and R. Buyya, "MRPGA: an extension of MapReduce for parallelizing genetic algorithms," in *IEEE Fourth International Conference on eScience, eScience'08*, pp. 214–221, IEEE, 2008.

[22] A. Verma, X. Llora, D. E. Goldberg, and R. H. Campbell, "Scaling genetic algorithms using MapReduce," in *Ninth International Conference on Intelligent Systems Design and Applications, ISDA'09*, pp. 13–18, IEEE, 2009.

[23] A. Radenski, "Distributed simulated annealing with MapReduce," in *Applications of Evolutionary Computation*, pp. 466–476, Springer, 2012.

[24] W.-P. Lee, Y.-T. Hsiao, and W.-C. Hwang, "Designing a parallel evolutionary algorithm for inferring gene networks on the cloud computing environment," *BMC systems biology*, vol. 8, no. 1, p. 5, 2014.

[25] C. Zhou, "Fast parallelization of differential evolution algorithm using MapReduce," in *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, pp. 1113–1114, ACM, 2010.

[26] K. Tagawa and T. Ishimizu, "Concurrent differential evolution based on MapReduce," *International Journal of Computers*, vol. 4, no. 4, pp. 161–168, 2010.

[27] M. Daoudi, S. Hamena, Z. Benmounah, and M. Batouche, "Parallel differential evolution clustering algorithm based on MapReduce," in *6th International Conference of Soft Computing and Pattern Recognition (SoCPaR)*, pp. 337–341, IEEE, 2014.

[28] K. Govindarajan, D. Boulanger, V. S. Kumar, and Kinshuk, "Parallel particle swarm optimization (ppso) clustering for learning analytics," in *2015 IEEE International Conference on Big Data (Big Data)*, pp. 1461–1465, Oct 2015.

[29] C.-W. Tsai, S.-J. Liu, and Y.-C. Wang, "A parallel metaheuristic data clustering framework for cloud," *Journal of Parallel and Distributed Computing*, vol. 116, pp. 39–49, 2018.

[30] R. Qi, Z. Wang, and S. Li, "Pairwise test generation based on parallel genetic algorithm with spark," in *Proceedings of the International Conference on Computer Information Systems and Industrial Applications*, pp. 67–70, 2015.

[31] L. Wang, Y. Wang, and Y. Xie, "Implementation of a parallel algorithm based on a spark cloud computing platform," *Algorithms*, vol. 8, no. 3, pp. 407–414, 2015.

[32] C.-W. Tsai, H.-C. Chang, K.-C. Hu, and M.-C. Chiang, "Parallel coral reef algorithm for solving jsp on spark," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 001872–001877, Oct 2016.

[33] J. Garcia Conejeros, F. Altimiras, A. Peña Fritz, G. Astorga, and O. Peredo, "A binary cuckoo search big data algorithm applied to large-scale crew scheduling problems," *Complexity*, 05 2018.

[34] C. Deng, X. Tan, X. Dong, and Y. Tan, "A parallel version of differential evolution based on resilient distributed datasets model," in *Bio-Inspired Computing-Theories and Applications*, pp. 84–93, Springer, 2015.

Cloud Computing, Big Data y las Arquitecturas de Referencia para la Industria 4.0

Nancy Velásquez [1], Elsa Estevez[2,3] y Patricia Pesado[1,4]

[1] *Facultad de Informática, Universidad Nacional de La Plata (UNLP), Argentina,*
[2] *Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur (UNS), Argentina*
[3] *Instituto de Ciencias e Ingeniería de la Computación, UNS-CONICET, Argentina*
4 Instituto de Investigaciones en Informática-LIDI, Facultad de Informática, UNLP, Argentina
nancy.velasquez87@hotmail.com, ece@cs.uns.edu.ar, ppesado@lidi.info.unlp.edu.ar

Abstract

The Industry 4.0 promotes the use of Information and Communication Technologies (ICT) in manufacturing processes to obtain customized products satisfying demanding needs of new consumers. The Industry 4.0 approach transforms the traditional pyramid model of automation to a network model of interconnected services. This new model allows the creation of ecosystems to make the production process more flexible through connecting systems and sharing data. In this context, cloud computing and big data are critical technologies for leveraging the approach. Thus, this paper analyzes cloud computing and big data under the lenses of two leading reference architectures for implementing Industry 4.0: 1) the Industrial Internet Reference Architecture (IIRA) of the Internet Consortium (IIC), and 2) the Reference Architecture Model for Industry 4.0 (RAMI 4.0). A main contribution of this paper is discussing needs, benefits, and challenges of applying cloud computing and big data in the Industry 4.0.

Keywords: Industry 4.0; Industrial Internet Reference Architecture (IIRA); Reference Architecture Model for Industry 4.0 (RAMI 4.0); Cloud Computing; Big Data.

1. Introducción

La Industria 4.0 (I4.0) es un término acuñado en Alemania, denominado *Cuarta Revolución Industrial* - la primera revolución la marcó el uso de la máquina de vapor, la segunda la energía eléctrica y la tercera la electrónica. Cada una generó cambios en la forma de producción así como en la sociedad. La cuarta revolución se caracteriza por la convergencia del mundo real con el mundo digital y la unión de la Tecnología de Operaciones (OT, Operation Technology) con la Tecnología de Información (TI).

Considerando la importancia que representa para las industrias adoptar nuevos métodos de producción mediante la utilización de los beneficios de la Tecnología de la Información y Comunicaciones (TIC) e incidir en la economía, existen actualmente arquitecturas de referencia que facilitan su adopción y continua evolución [1]. Por ejemplo, en Alemania, se publica el *Reference Architecture Management Industrial 4.0* (RAMI 4.0) por medio del proyecto denominado *Platform Industrie 4.0* cuyo objetivo es el desarrollo e implementación de la Industria 4.0 como estrategia del Gobierno Federal alemán; iniciativa a la cual se suman Italia, Francia, y España, entre otros países. Debido al impulso gubernamental, actualmente RAMI 4.0 se convirtió en la norma alemana DIN SPEC 91345[2]. En Estados Unidos, empresas como General Electric, AT&T, Cisco, IBM e Intel fundan el *Industrial Internet Consortium* (IIC) con el objetivo de impulsar la implementación del Internet Industrial para la interconexión de máquinas y dispositivos mediante la arquitectura de referencia denominada *Industrial Internet Reference Architecture* (IIRA) [3]. El objetivo principal de RAMI 4.0 e IIRA es digitalizar los procesos de fabricación para dotarlos de "inteligencia" mediante el empleo de la tecnología, posibilitando el cambio en el modelo de producción actual [4].

En este artículo inicialmente se presenta un breve resumen de los fundamentos de la Industria 4.0 y de computación en la nube (cloud computing) y grandes volúmenes de datos (big data) como sus habilitadores tecnológicos (Sección 2). A continuación, se explican las arquitecturas de referencia: *RAMI 4.0* de Alemania, e *IIRA* de Estados Unidos (Sección 3), y se realiza una comparación de las capas relacionadas con el almacenamiento de datos en estos dos modelos (Sección 4). Finalmente, se discuten retos, oportunidades, trabajos futuros e investigaciones asociados a esta temática (Sección 5).

2. Fundamentos

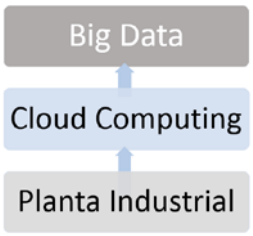
La Industria 4.0 transforma el proceso de producción tradicional de forma piramidal basado en la Norma ISA 99/IEC 62443 hacia un modelo de trabajo en red, totalmente interconectado por medio del Internet de

las Cosas (IoT) [5], facilita la flexibilidad de la producción de acuerdo con la demanda, la optimización de recursos, la reducción de costos y el establecimiento de comunicación efectiva entre los clientes, la industria y los proveedores [6]. Incorpora los beneficios de las TI por medio de los denominados habilitadores tecnológicos como: la simulación, inteligencia artificial, robótica colaborativa, realidad virtual y aumentada, ciberseguridad, integración vertical y horizontal, impresión 3D, cloud computing y big data, posibilitando nuevos procesos, nuevos productos y nuevos modelos de negocio

Adicionalmente, el desarrollo tecnológico actual permite disponer de componentes electrónicos cada vez más pequeños, con mayor capacidad y a menor costo y facilita la incorporación de sensores en los productos y en las máquinas. Esta característica incide en el proceso de producción, el producto y el modelo de negocio. En particular, la incorporación de dichas tecnologías al proceso de desarrollo de productos dota de inteligencia al proceso y obtiene información de las diferentes fases, como: el diseño (desde la idea original), la fabricación (ciclo de vida), la logística, distribución, entrega y post venta. Asimismo, permite incluir en el diseño las especificaciones del producto solicitadas por el cliente y obtener un producto personalizado, realizar el diseño de forma colaborativa sin importar la ubicación física de los integrantes, realizar simulaciones antes de pasar a producción, planificar la producción de manera flexible, con tiempos de respuesta según los productos a fabricar, y permitir la producción colaborativa en diferentes ubicaciones geográficas [7]. Del mismo modo, la Industria 4.0 promueve realizar la trazabilidad completa del ciclo de vida del producto, integrar horizontal y verticalmente las aplicaciones, disponer de información en tiempo real para la toma de decisiones oportunas, incluir información para toma de decisiones autónomas sin participación humana, contar con información transparente durante todo el proceso de producción, detectar errores en la fabricación y regresar inmediatamente para corregirlos, detectar posibles fallas y actuar preventivamente. Por último, el enfoque también permite planificar el mantenimiento preventivo, monitorear el proceso de manera remota para evitar cortes en la producción por fallas de los equipos, planificar la fabricación en diferentes puntos sin importar la ubicación geográfica, facilitar la comunicación con diferentes proveedores durante todo el proceso, reducir los stocks y los plazos de entrega, obtener información del uso y vida del producto y disponer de información para generar nuevos modelos de negocio.

La "hiper-conectividad" de los diversos sistemas involucrados en el proceso de producción, en la cual se basa la Industria 4.0, multiplica los datos que se generan en tiempo real de diferentes fuentes y variados formatos que deben ser almacenados y analizados con tecnologías y algoritmos avanzados. Por este motivo, es que se utiliza la computación en la nube (cloud computing) y las técnicas asociadas a los grandes volúmenes de datos (big data). Ambos se consideran requisitos fundamentales para la Industria 4.0, proporcionando componentes relevantes y necesarias para la implementación del enfoque [8]. La relación existente entre estos dos habilitadores y la planta industrial se presenta en la Figura 1.

Figura 1 Relación Cloud Computing y Big Data



2.1. Cloud Computing

Cloud Computing permite almacenar grandes cantidades de datos. Dicha capacidad es principalmente importante para almacenar los datos generados durante todo un proceso de producción, considerando que las máquinas y los sensores producen mayor cantidad de datos que una persona y están siempre conectados. Asimismo, cloud computing permite disminuir la inversión en recursos tecnológicos permitiendo contratar bajo demanda el espacio de almacenamiento y la capacidad de procesamiento, lo que brinda flexibilidad, agilidad y adaptabilidad [9]. Mediante una estructura escalable posibilita el consumo de recursos a solicitud. Esto permite reducir costos, ya que evita adquirir, servidores, licencias, y personal especializado para el mantenimiento. Asimismo, permite el ahorro de energía; brinda facilidad de acceso al almacenamiento desde diversos puntos geográficos, diferentes horarios, independientemente de la plataforma y dispositivo de conexión. Todo esto, facilita la creación de ecosistemas de fabricación, fomenta la colaboración entre clientes y proveedores. En particular, el cliente puede participar en todas las etapas del proceso de producción y por consiguiente se mejora su satisfacción.

Debido a los grandes volúmenes de datos procesados en la nube, existe tiempo de latencia que puede afectar el desempeño de la producción, para esto se puede utilizar el "*Fog Computing*" (computación en "la niebla"). Fog computing se refiere a la capacidad de procesamiento para análisis de datos y funciones de control en base a grandes volúmenes de datos recibidos por dispositivos de Internet de las Cosas

(IoT). Los nodos que realizan este procesamiento están ubicados generalmente en la misma red local que los dispositivos de IoT. Habitualmente, consta de capacidad de almacenamiento y computación. Por estos motivos, el fog computing, comparado con el coud computing, disminuye la inestabilidad, evita la congestión y mejora la calidad de servicio [10]. Adicionalmente, para reforzar la seguridad y privacidad de los datos existe la posibilidad de implementar nubes privadas [11] para uso exclusivo de industrias que necesitan extrema confidencialidad.

El Comité ISO/IEC JTC 1/SC 38 (IEC 19944) es el responsable de la estandarización de cloud computing [12], y provee modelos y estándares para las pequeñas y medianas empresas (PyMEs) [13].

2.2. Big Data

Técnicas de big data permiten analizar el enorme volumen de información generado en un ecosistema de producción de la Industria 4.0 utilizando análisis avanzado, histórico, predictivo [14] y descriptivo del estado y funcionamiento de las máquinas involucradas en los procesos de producción. El análisis de datos para mantenimiento predictivo permite reducir ineficiencias y costos, anticipando fallas en equipos y permitiendo dar mejores respuestas a situaciones emergentes y remotas causadas por diferentes factores como mal clima, alta humedad, temperatura elevada, exposición a gases, etc. Como otro ejemplo, el análisis predictivo también permite realizar el control de rutas determinando donde se generan disturbios o existe un mal estado de las vías. Asimismo, permite determinar patrones y correlacionar datos para la mejora de los procesos industriales, tomar decisiones en tiempo real para la eficiencia operativa, definir nuevos modelos de negocio basado en el comportamiento del cliente, el diseño de nuevos servicios asociados a los productos existentes [14] y el aprendizaje automático. En resumen, las técnicas de big data permiten generar valor a la cadena de producción a partir del uso inteligente de los datos [15].

El uso de big data no depende del tamaño de la industria, depende de la estrategia empresarial que se enfoca en obtener provecho de los datos para mejorar el proceso de fabricación y determinar los productos que el mercado demanda para satisfacer al cliente. Esto permite cambiar la forma actual en la cual la empresa investiga, analiza y decide el producto a sacar al mercado, incrementando las posibilidades que el producto sea vendido con éxito. Asimismo, otra área importante de aplicación de big data es el campo de la seguridad, para análisis y correlación de datos para detectar fraudes, y realizar análisis de comportamiento de clientes, entre otros.

Dada la relevancia de big data para la Industria 4.0, el Comité ISO/IEC JTC 1/WG 9 está elaborando la norma para su aplicación en este contexto [12].

3. Arquitecturas de Referencia de la Industria 4.0

A nivel mundial existen esfuerzos para implementar la Industria 4.0. Para apoyar estos esfuerzos, existen dos arquitecturas de referencia importantes - la *Reference Architecture Model for Industry 4.0* (RAMI 4.0), y la *Industrial Internet Reference Architecture* (IIRA). En la mayoría de países europeos han tomado a RAMI 4.0 como la base para sus programas de desarrollo de la producción y la economía. IIRA tiene amplio despliegue por su origen, Estados Unidos, y su permanencia en el mercado. Adicionalmente, China también dispone de un modelo para su programa *Made in China 2015*. En base a su nivel de incidencia y relevancia, en las siguientes dos secciones se analizan RAMI 4.0 e IIRA como los principales modelos de arquitectura de referencia para la Industria 4.0.

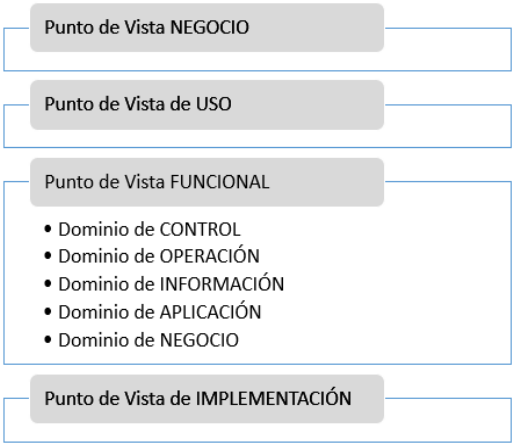
3.1. IIRA

El IIRA es un modelo de arquitectura abierta, basado en la norma internacional ISO/IEC/IEEE 42010:2011 y en estándares para sistemas IoT. Está orientado hacia múltiples industrias, como son la salud, transporte, energía, sector público y la manufactura, procurando la interoperabilidad de sistemas y procesos [16]. El centro de esta arquitectura son las cuatro capas denominadas *puntos de vista*: 1) *Punto de Vista del Negocio*, donde se genera información de los interesados (stakeholders); 2) *Punto de Vista de Uso*, donde se describe los requerimientos de uso; 3) *Punto de vista Funcional*, para los requerimientos funcionales del producto; y 4) *Punto de Vista de Implementación* para las consideraciones de los componentes funcionales y técnicos para la implementación.

El Punto de Vista Funcional, contiene cinco dominios: Control, Operación, Información, Aplicación y Negocio (ver Figura 2). Estos dominios establecen el flujo de datos y el control entre ellos. A saber, en el de Control se obtienen los datos de los sensores y de los elementos físicos del mundo real para realizar el control inmediato de los mismos; en el de Operación, se procesa la información de los sistemas de control industrial; en el de Información se recopilan los datos de otros dominios para el análisis de todo el sistema; en el de Aplicación se trata la información de alto nivel para la optimización global,

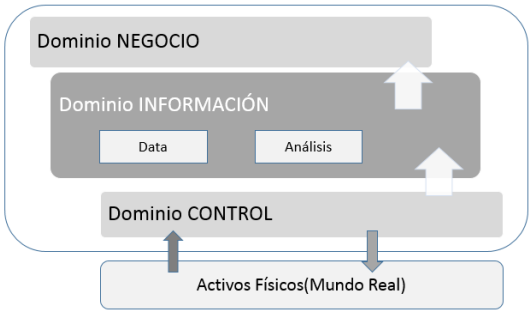
y el de Negocio se ocupa de la información para la integración con sistemas existentes y nuevos.

Figura 2 Puntos de Vista de IIRA



El Dominio de Información recopila los datos de los otros dominios, en especial del dominio de control, con el objetivo de analizarlos para dotar al sistema de inteligencia, que se revierta en información para la toma de decisiones, incluso en tiempo real, optimización de las operaciones y mejoramiento del sistema. Los datos aquí procesados alimentan al Dominio del Negocio (ver Figura 3).

Figura 3. Dominio de Información - Flujo de Datos



El volumen de datos de los procesos industriales digitalizados demanda una extraordinaria capacidad de cómputo, facilidades de acceso desde cualquier lugar y diferentes tipos de dispositivos, alta disponibilidad y flexibilidad, razón por la cual cloud computing está considerado como un habilitador de la Industria 4.0. De acuerdo con la demanda de recursos de cómputo y almacenamiento, las industrias pueden elegir entre el servicio de Infraestructura como Servicio (IaaS), Plataforma como Servicio (PaaS) o Software como Servicio (SaaS). La adopción del cloud computing en entornos de fabricación se extiende debido a la facilidad de configurar redes públicas, privadas o híbridas para proteger la confidencialidad de la información, según amerite.

Los dispositivos físicos, como los sensores, cada vez están dotados de mayores recursos de cómputo, lo que determina que varíe la capacidad de procesamiento que requiere el Dominio de Control, situación que no afecta a los datos y al análisis que se debe realizar en el Dominio de Información. Asimismo, como una medida de contingencia, el Dominio de Información debe estar en el exterior de la empresa, situación que cloud computing también facilita.

Por último, el crecimiento del volumen de datos de los sistemas de Industrial Internet of Things (IIoT) es constante, así como su complejidad. Por eso, requiere el uso de enfoques analíticos avanzados de big data para correlacionar la información del ciclo de vida del producto, que incluye la postventa [16]. La conectividad en IIRA se realiza mediante el Connectivity Core Standard [17].

3.2. RAMI 4.0

La arquitectura RAMI 4.0 combina aspectos relacionados al proceso de fabricación, al producto y a las TI mediante una estructura tridimensional jerárquica orientada al servicio.

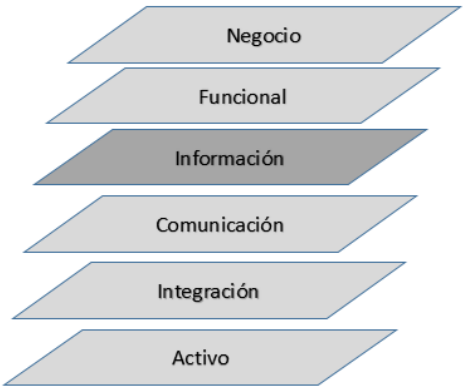
En esta arquitectura, el eje uno representa la jerarquía de la fabricación y se basa en la norma IEC 62264 /IEC 61512. En este eje se inicia el proceso de producción de los productos inteligentes, e incluye la fábrica inteligente (campo de dispositivo, control de dispositivo, estación, centros de trabajo, empresa) y el mundo conectado.

El eje dos corresponde al producto, y está basado en la norma internacional IEC 62890. Este eje está relacionado con el ciclo de vida del producto, comprendiendo el desarrollo, las ventas, el soporte después de la venta y la obsolescencia. En RAMI 4.0 se representa el ciclo de vida del producto mediante el tipo y la instancia [18]. El tipo comprende el desarrollo (plan de construcción, incluido el desarrollo, construcción, prototipo, pruebas, etc.) y el mantenimiento para el uso (actualización, mantenimiento, etc.). La instancia abarca a la producción (producto) y su mantenimiento (gestión de instalación, incluido el servicio, mantenimiento, reciclaje, etc.).

El eje tres denominado de la Arquitectura y relacionado con la Tecnología de la Información comprende seis niveles denominados: Activos, Integración, Comunicación, Información, Funcional y de Negocio. El nivel de Activos contiene información digital de los elementos físicos del mundo real. El nivel de Integración es el nexo entre el mundo real y el digital. Los siguientes cuatro niveles corresponden al mundo digital [19]. El nivel

de Comunicación comprende el acceso a la información y las funciones de un activo conectado con otros, usa un formato específico con información sobre los datos que se usan, donde se usan y como se distribuyen. El nivel de Información comprende los datos relacionados con la funcionalidad del activo y del lote en tiempo real. Esta información puede impactar en el siguiente nivel. El nivel Funcional describe las funciones lógicas y técnicas de un activo y de la plataforma para la integración con otros activos. Por último, el nivel de Negocio contiene información de la organización, procesos comerciales, la visión del negocio y aspectos legales [20] (ver Figura 4).

Figura 4. RAMI 4.0 – Eje de la Arquitectura. Niveles de TI



La arquitectura de referencia RAMI 4.0 es una red de componentes interconectados todos entre todos mediante el "*Administration Shell*" que contiene los datos de la representación virtual de un objeto del mundo real con sus características y funcionalidades propias [21].

En la arquitectura RAMI 4.0 existe un intercambio constante de datos que generan grandes volúmenes de información que, preferentemente, deben ser almacenados en la nube y analizados por técnicas de big data [22]. La comunicación se establece por medio estándar *OPC Unified Architecture* (OPC-UA) – un estándar para intercambio de información para comunicación industrial [23].

4. Comparación entre IIRA y RAMI 4.0

En el 2015, representantes de Plattform I4.0 y el IIC unieron esfuerzos para identificar aspectos en común de las dos arquitecturas de referencia IIRA y RAMI 4.0 concluyendo que presentan más similitudes que diferencias y pueden ser complementarias. RAMI 4.0 se enfoca en fabricar inteligentemente mientras que IIRA en construir, desplegar y operar grandes sistemas conectados. A modo de ejemplo, partes y piezas fabricadas de acuerdo con RAMI 4.0, son elementos de los sistemas operados por IIRA. Esta característica sirve para resaltar la importancia de cloud computing y big data en la Industria 4.0, independiente del modelo.

Las dos arquitecturas son similares porque se centran en la interoperabilidad, consideran como punto de partida los activos y los niveles del eje de arquitectura de RAMI 4.0 son compatibles con los dominios funcionales de IIRA como se muestra en la Tabla 1.

Tabla 1 Comparación IIRA y RAMI 4.0

IIRA **Dominios Funcionales**	RAMI 4.0 Niveles de Arquitectura
Sistemas físicos	Nivel de Activos
Dominio de Control	Nivel de Integración Nivel Funcional Nivel de Comunicación
Dominio de Información. Recopila los datos de los otros dominios en especial del Dominio de Control, para su análisis. Transforma y analiza los datos El volumen de datos es creciente y requiere de cloud computing y big data.	Nivel de Información. Contiene los datos de la funcionalidad del activo en tiempo real y en lote. La información de tiempo real impacta al siguiente nivel, pero es en este nivel donde se la almacena y analiza, siendo necesario el uso del cloud computing y big data.
Dominio de Operación y Aplicación	Nivel Funcional
Dominio del Negocio	Nivel de Negocio

Los sistemas físicos y el nivel de activos comprenden los componentes físicos del mundo real, los documentos, el software y personas. El dominio de control obtiene los datos de los activos para controlarlos inmediatamente y de manera similar en el nivel de integración se relaciona la información del activo real con la información digital. El Dominio de Información en IIRA y el Nivel de Información en RAMI 4.0 tienen funcionalidad similar, concentran los datos y la funcionalidad técnica de los activos, como pueden ser sensores, máquinas, herramientas, aplicaciones, etc., se encargan de analizar los datos para proporcionar inteligencia al sistema [4]. El flujo de datos de estos niveles con sus adyacentes se caracterizan por la variedad, la integridad y la integración, siendo relevante el uso del cloud computing y big data [24]. Cabe desatacar que en el reporte técnico del Industrial Internet Consortium publicado en diciembre de 2017 [4] consta que "la comunicación de nube a nube y de empresa a nube está aún en discusión dentro de la comunidad de la Industria 4.0".

En este contexto, empresas alemanas como Bosch, Siemens y Deutsche Telekom han emprendido en

proyectos para implementar la Industria 4.0 y también ofrecen portafolios de soluciones de Industria 4.0. Volkswagen, Bayer, Mercedez Benz, BMW tienen en marcha proyectos de Industria 4.0; IBM, SAP, Microsoft, Deutsche Telecom, Oracle, Software AG y otras ofrecen plataformas de integración en la nube como servicio; Bosch SI, IBM, SAP, Microsoft, Deutsche Telekom, Cisco, entre otras ofrecen soluciones de software, soluciones big data y de big data industrial para recopilación y tratamiento de datos masivos de activos industriales [25]. Por su parte, Adidas está implementando este enfoque en su planta de México [26]. Cabe destacar que Bosch presentó en el 2016, en la conferencia "Bosh ConnectedWorld IoT, en Berlín un proyecto para su planta de Homburg en Alemania, para gestionar y optimizar la producción de una válvula hidráulica que permita reducir el consumo eléctrico utilizando las dos arquitecturas de referencia: RAMI 4.0 e IIRA [27].

5. Conclusiones

La Industria 4.0 digitaliza la producción mediante la representación del mundo real y la interconectividad de sus actores. En este proceso, se generan datos en tiempo real de diferentes fuentes y de diversos tipos, que generan grandes volúmenes de datos. Estos datos deben ser almacenados y analizados a fin de crear valor para la industria para satisfacer a un mercado globalizado y competitivo que demanda productos personalizados.

El campo de investigación sobre las TI que habilitan la Industria 4.0 es muy amplio y cambiante. Asimismo, depende del surgimiento de tecnologías disruptivas que se relacionen con la fabricación. Sin embargo, dos pilares tecnológicos clave para la Industria 4.0 son: 1) cloud computing – para facilitar el almacenamiento de los datos de producción de forma tal que se soporte y permita la inter-conectividad de los procesos de producción con proveedores y clientes; y 2) big data – da soporte al análisis avanzado de la información recolectada que crece en tamaño y complejidad. El uso de estas tecnologías es independientemente del tamaño de la industria y está de acuerdo con la estrategia empresarial de innovación y enfoque en el cliente.

El uso de cloud computing y big data para la Industria 4.0 abre un abanico de oportunidades para futuros trabajos de investigación. Los mismos se discuten a continuación.

El desarrollo de big data y su relación con temas de seguridad de la información y de ciberseguridad es aún muy incipiente y presenta grandes desafíos. Un tema sumamente relevante para investigaciones futuras es la seguridad de la información administrada por los procesos de la Industria 4.0, como así también cuestiones relacionadas a la propiedad de la información, el control de acceso, la identidad digital, criptografía, y cifrado, entre otras. De manera similar, el alto grado de interconexión de la Industria 4.0 requiere de estándares para facilitar su implementación. Por esto, se debe profundizar el estudio de los estándares existentes actualmente sobre cloud computing y big data, para identificar cuáles deben ser adaptados y cuales se deben definir para hablar el mismo idioma. El comité ISO/IEC JTC 1/SC 38 Cloud Computing and Distributed Platforms está elaborando estándares para usar el cloud computing para el almacenamiento de datos y la comunicación entre máquinas y humanos. El comité ISO/IEC JTC 1/WG 9 Big Data está trabajando en la elaboración de una norma para definir la arquitectura que permita realizar el uso eficiente del Big Data, evaluar la recopilación de datos no estructurados para optimizar los procesos de producción y logística. Aspectos como sistemas de gestión de continuidad del negocio, ciberseguridad de los sistemas de control y automatización industrial, evaluación del riesgo de seguridad y diseño del sistema, requisitos de desarrollo del producto, robótica, también deben ser estandarizados [28]. Asimismo, actualmente está en discusión la comunicación de cloud a cloud, y de empresa a cloud, ofreciendo oportunidades para investigaciones futuras. Por último, es necesario contar con profesionales calificados en estas técnicas. Por eso, el desarrollo de programas de formación de recursos humanos especialistas en temas de cloud computing y big data para la Industria 4.0 es crítico.

Referencias

[1] A. T. Kearney, "Readiness for the Future of Production : Country Profiles," 2017.

[2] Plattform Industrie 4.0, "Digitization of Industrie – Plattform Industrie 4.0," no. April, pp. 6–9, 2016.

[3] S. Schrecker *et al.*, "Industrial Internet of Things Volume G4 : Security Framework," *Ind. Internet Consort.*, pp. 1–173, 2016.

[4] A. Industrial, I. Consortium, and Platform Industrie 4.0, "Architecture Alignment and Interoperability," 2017.

[5] Plattform Industrie 4.0, "Digitization of Industrie – Platform Industrie 4.0," *Plattf. Ind. 4.0*, no. April, p. 28, 2016.

[6] I. M. S. Board and F. I. for M. E. and A.

IPA, “Factory of the future,” 2015.
[7] R. G. White and P. Ramachandran, “in a Global Context,” pp. 257–283, 2013.
[8] Fundación Telefónica, “La transformación digital de la industria española. Informe preliminar.,” *Ind. Conectada 4.0*, p. 120, 2015.
[9] L. Thames and D. Schaefer, “Software-defined Cloud Manufacturing for Industry 4.0,” *Procedia CIRP*, vol. 52, pp. 12–17, 2016.
[10] A. Gilchrist, *Industry 4.0 The Industrial Internet of Things*, 2016th ed. Bangken, Nonthaburi, Thailand, 2016.
[11] D. Protection, “Data Protection & Industrie 4.0.”
[12] G. Lars Adolph, Bundesanstalt, “DIN / DKE – Roadmap G E R M A N S TA N D A R D I Z AT I O N.”
[13] P. G. Morales, J. A. A. España, J. E. G. Zárate, C. C. O. González, and T. E. R. Frías, “La Nube Al Servicio De Las Pymes En Dirección a La Industria 4.0,” *Pist. Educ.*, vol. 39, no. 126, pp. 85–98, 2017.
[14] J. Lee, B. Bagheri, and H.-A. Kao, “Recent Advances and Trends of Cyber-Physical Systems and Big Data Analytics in Industrial Informatics,” *Int. Conf. Ind. Informatics*, no. November 2015, pp. 1–6, 2014.
[15] M. C. Vega, P. O. Vivas, C. M. Ríos, C. G. Luis, B. C. Martín, and A. H. Seco, “las tecnologías IoT,” Madrid, 2015.
[16] S.-W. (Thingswise/Intel) Lin *et al.*, “Reference Architecture,” *Ind. Internet Consort.*, vol. 1.80, no. November, pp. 1–7, 2017.
[17] A. Industrial, I. Consortium, R. Neubert, and H. Munz, “Architecture Alignment and Interoperability.”
[18] P. Adolphs, “Rami 4.0,” 2015, no. June.
[19] Dr. Kartsen Schweichhart, “Reference Architectural Model Industrie 4.0 (RAMI 4.0): An Introduction,” in *Publikationen der Plattform Industrie 4.0*, 2016, vol. 0, no. April.
[20] M. Nardello, C. Møller, and J. Gøtze, “Organizational Learning Supported by Reference Architecture Models : Industry 4 . 0 Laboratory Study,” no. 12, pp. 22–38, 2017.
[21] Federal Ministry for Economic Affairs and Energy (BMWi), “Structure of the Administration Shell,” p. 52, 2015.
[22] S. Yin and O. Kaynak, “Big Data for Modern Industry: Challenges and Trends,” *Proc. IEEE*, vol. 103, no. 2, pp. 143–146, 2015.
[23] O. P. C. Ua, “Industrie 4.0 Communication Guideline.”
[24] M. Hankel and B. Rexroth, “The Reference Architectural Model Industrie 4.0 (RAMI 4.0),” *ZWEI Die Elektroind.*, vol. 1, no. April, pp. 1–2, 2015.
[25] D. Migu and O. Econ, “La industria 4.0 en Alemania,” 2017.
[26] G. Figueroa, “Adidas se suma a la industria 4.0 de la mano de Siemens,” Mexico, pp. 4–5, 28-Apr-2017.
[27] infoplc, “Bosch y un proyecto con RAMI 4.0 e IIRA,” 2016. [Online]. Available: http://www.infoplc.net/noticias/item/103262-bosch-connectedworld-iot.
[28] UNE, “Estandarización,” 2017.

Orquestación de servicios para el desarrollo de aplicaciones para big data

Maria Murazzo[1], Miguel Guevara[1], Martin Tello[1], Nelson Rodriguez[1], Fabiana Piccoli[2], Mónica Gimenez[3]

[1]*Departamento de Informática, F. C. E. F. y N, Universidad Nacional de San Juan*
marite@unsj-cuim.edu.ar, migueljoseguevaratencio@gmail.com, martinl.tello@gmail.com, nelson@iinfo.unsj.edu.ar

[2] *Departamento de Informática - F.C.F.M. y N, Universidad Nacional de San Luis*
mpiccoli@unsl.edu.ar

[3]*Facultad de Ciencias Exactas Físicas y Naturales – Universidad Nacional de La Rioja*
monik.gimenez@gmail.com

Abstract

Los avances tecnológicos han permitido que se generen grandes cantidades de datos, los cuales necesitan ser almacenados y procesados de manera eficiente. Surge así el paradigma Big Data, donde el principal requerimiento no solo es la capacidad de cómputo, sino el manejo en un tiempo razonable de ingentes cantidades de datos. En este contexto, las aplicaciones para big data necesitan ser escalables, livianas, autocontenidas, distribuidas y replicadas con el objetivo de lograr la mejor performance frente a variaciones del volumen de datos. Para lograr esto, este trabajo propone ajustar la construcción de aplicaciones a una arquitectura basada en microservicios los cuales puedan ser implementados con contenedores. La replicación y distribución para lograr altos niveles de escalabilidad se plantea mediante la orquestación de contenedores sobre una arquitectura distribuida virtualizada.

Keywords: Big Data, Orquestación, Microservicios, Contenedores, Docker

1. INTRODUCCIÓN

La integración de Cloud Computing (CC) [1] con Internet of Thing (IoT) [2] representa el próximo gran paso en las TI del futuro. Las aplicaciones que surjan de esta integración abrirán nuevas perspectivas de negocio y oportunidades de investigación. Gracias al paradigma CloudIoT [3], la vida cotidiana mejorará, por ejemplo, en las Smart city [4], permitirán servicios públicos más eficientes través de aplicaciones ubicuas que mejorarán la calidad de vida de los ciudadanos. El futuro de este paradigma pasa por la convergencia hacia una plataforma de servicio común que supere los desafíos planteados por la heterogeneidad de los dispositivos y las tecnologías involucradas. Esto conlleva tener en cuenta la ubicuidad y la omnipresencia de los dispositivos, lo cual, requieren plataformas de computación escalables.

Esta convergencia ha provocado que se creen datos a una tasa exponencial, dando lugar al paradigma Big Data [5]. Big data se caracteriza por tres aspectos: a) el volumen de los datos, b) los datos no se pueden almacenar de forma tradicional y c) los datos are generados, capturados y procesados con rapidez. Estas características hacen que los sistemas de cómputo convencionales sean inapropiados para lograr una gestión adecuada del big data [6].

Las arquitecturas de software tradicionales no permiten almacenar y gestionar grandes volúmenes de datos con el fin de extraer un valor estratégico. El reto del big data es convertir grandes volúmenes de datos en inteligencia de negocio, lo cual implica una gestión de datos distinta, en la que se incorpora el análisis en tiempo real para orientar la toma de decisiones. No se trata de recopilar datos y hacer análisis detallados, sino más bien en hacer interpretaciones rápidas que orienten en la toma de decisiones a partir de datos de cualquier fuente [7].

Construir aplicaciones para big data tiene como principal problema la necesidad de administrar recursos, lo cual implica analizar y planificar la forma en la cual la aplicación debe escalar.

Tradicionalmente, la construcción de aplicaciones ha seguido un paradigma de escalado vertical mediante el cual la solución a los problemas de cuellos de botellas se resuelve mediante el

aprovisionamiento de mayor cantidad de recursos. Pero cuando se debe procesar y analizar grandes cantidades de datos, las aplicaciones deben pensarse y construirse de forma distribuida; obligando a contar con mecanismos de *escalabilidad horizontal*, que permita distribuir la carga de trabajo dinámicamente y paralelizar las tareas. La escalabilidad horizontal implica balanceo de carga, replicación de recursos, reinstanciación de recursos en tiempo real y optimización en su uso, de forma que el usuario no perciba degradación de performance.

El presente trabajo tiene como objetivo definir una manera de construir aplicaciones para big data que escalen de forma horizontal, lo cual asegurará que los datos sean almacenados, procesados y visualizados en forma satisfactoria.

El resto del trabajo se organiza de la siguiente manera: en la próxima sección se explican las generalidades de las arquitecturas de soporte tanto de hardware como de software. En la Sección 3 se explican las generalidades de la orquestación de servicios. En la Sección 4 se realiza una breve descripción de la herramienta seleccionada. En la Sección 5 se describe el caso de estudio abordado. En la sección final se abordan las conclusiones y futuros trabajos.

2. ARQUITECTURA DE SOPORTE

Como se mencionó en la sección anterior, las tecnologías, arquitecturas, lenguajes y metodologías tradicionales son inapropiadas para realizar el tratamiento de datos masivos. Es por ello que una alternativa es usar una arquitectura distribuida [8] con el fin de proveer tolerancia a fallos, distribución y replicación.

La computación distribuida es la evolución de los sistemas de cómputo convencional, los cuales permiten realizar operaciones de cómputo intensivo y mejorar la velocidad de procesamiento; involucrando tecnologías tal como cluster y cloud [9].

Los entornos distribuidos son ideales para ejecutar aplicaciones computacionalmente costosas con manejo de grandes cantidades de datos, a fin de lograr resultados en menor tiempo.

La conjunción de big data y la computación distribuida, se enfoca en la paralelización del problema mediante la distribución de los datos y la delegación del cómputo en los nodos con capacidad de procesamiento [10].

Construir aplicaciones capaces de almacenar, procesar y visualizar grandes volúmenes de datos sobre una arquitectura distribuida implica la definición de los métodos de distribución y replicación a fin de lograr la escalabilidad deseada [11].

2.1. ARQUITECTURA DE SOFTWARE

Para construir aplicaciones con las características de escalado mencionadas en el apartado anterior, es necesario que la arquitectura de software se alinee con estas características.

En tal sentido, la arquitectura de microservicios [12] permite construir aplicaciones distribuidas como un conjunto de pequeños servicios desacoplados, desplegados de manera independiente y que trabajen coordinadamente.

De la misma manera que en una aplicación monolítica, los microservicios pueden acceder a datos a través de capas de persistencia, drivers o configuración específica para cada caso, exponiendo sus funcionalidades a través de API's REST y comunicándose entre sí con mecanismos ligeros tales como API de HTTP o mensajería. Una de las ventajas más importantes, es que los microservicios deben ser creados, mantenidos, ejecutados y distribuidos de forma totalmente independiente [13].

Una de las características más importantes que provee los microservicios es la escalabilidad [14]. Debido a que los microservicios se desarrollan en forma independiente uno de otros, también escalan en forma independiente, esto es muy importante cuando se trabaja con grandes volúmenes de datos pues permite la implementación de carga compartida de trabajo.

Los microservicios dependen no solo de la tecnología que se está configurando, sino también de una organización que tenga la cultura, el conocimiento y las estructuras establecidas para que los equipos de desarrollo puedan adoptar este modelo. Los microservicios forman parte de un cambio más amplio en los departamentos de TI hacia una cultura DevOps [15], en la que los equipos de desarrollo y operaciones trabajan estrechamente para respaldar una aplicación a lo largo de su ciclo de vida y pasan por un ciclo de publicación rápido o incluso continuo en lugar de un ciclo tradicional largo.

Un enfoque de microservicios también puede facilitar que los desarrolladores de aplicaciones ofrezcan interfaces alternativas a sus aplicaciones. Cuando todo es una API, las comunicaciones entre los componentes de la aplicación se estandarizan. Todo lo que tiene que hacer un componente para hacer uso de su aplicación y sus datos es poder autenticarse y comunicarse a través de esas API estándar [13]. Esto permite que tanto los que están dentro como, cuando corresponda, los que estén fuera de su organización, desarrollen fácilmente nuevas formas de utilizar los datos y servicios de su aplicación.

2.1. 1. CONTENERIZACIÓN

El uso de microservicios implica la construcción de aplicaciones a partir de múltiples componentes autosuficientes. Estos componentes pueden ser implementados con contenedores [16].

Un contenedor, es un proceso que, internamente, contiene la aplicación que se quiere ejecutar y todas sus dependencias usando indirectamente el kernel del sistema operativo que se esté usando. Los contenedores permiten implementar aplicaciones distribuidas debido a dos características: portabilidad, pues los contenedores se pueden ejecutar de forma independiente al hardware y al software donde se crearon; y baja sobrecarga, pues son livianos e introducen menos overhead que las VM (Virtual Machine) [17].

Realizando un paralelismo entre un contenedor y una VM (Virtual Machine) [18] [19], ambos son sistemas autocontenidos que tienen como principal diferencia que, una VM necesita contener todo el sistema operativo mientras un contenedor aprovecha el sistema operativo en el que se ejecute.

La principal ventaja [20] de utilizar contenedores es que no depende de las instalaciones del sistema anfitrión, es decir es posible realizar las instalaciones necesarias para desplegar aplicaciones dentro de un contenedor. Otra ventaja es que permite la portabilidad en las aplicaciones, esto quiere decir que además de crear contenedores, es posible definir repositorios que eventualmente alojarán a los contenedores con el objeto que otros usuarios los descarguen y los consuman como servicio.

Quizás la características más relevante de los contenedores es que permiten desplegar aplicaciones de forma rápida y escalables, ya que permite replicar ambientes de desarrollo, prueba y producción en poco tiempo, pudiendo aumentar o disminuir la cantidad de contenedores a medida que sean necesitados (escalabilidad elástica) [21].

El uso conjunto de contenedores y microservicios [22] mejora las capacidades de escalado, ya que el microservicio es portable y reutilizable; mientras que los contenedores proporcionan recursos eficientes, principalmente el empaquetado de aplicaciones y sus dependencias en un contenedor virtual que puede ejecutarse en cualquier servidor.

3. ORQUESTACIÓN DE SERVICIOS

Se ha dicho que cuando se trabaja con datos masivos es necesario escalar horizontalmente las aplicaciones, esto se puede lograr mediante el uso de múltiples nodos (cluster o cluster as a service). Para lograr esta forma de distribución es fundamental abstraerse de la complejidad de la plataforma subyacente, lo cual se logra mediante el uso de clusters de contenedor [23] como se muestra en la figura 1.

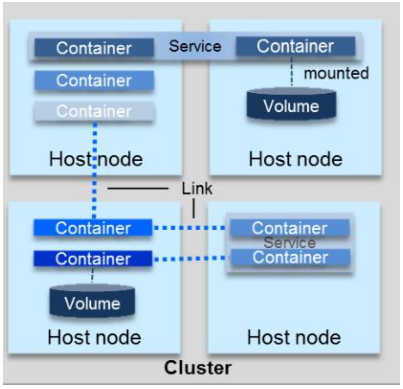


Figura 1: Arquitectura de un cluster de contenedores

Cada nodo en el cluster es un servidor virtual y puede tener múltiples contenedores con una semántica común, a su vez, una aplicación puede estar formada por un grupo lógico de varios contenedores, la cual puede escalar a través de múltiples nodos.

Esta forma de trabajar necesita administrar y coordinar los contenedores (microservicios), para ello es posible realizar orquestación o coreografía [24].

Orquestar significa que hay una entidad central (idealmente un microservicio en sí mismo) que coordina la comunicación entre los microservicios. Mientras que para microservicios coreografiados, cada servicio implicado en dicha coreografía "conoce" exactamente cuándo ejecutar sus operaciones y con quién debe interactuar, lo cual le quita transparencia e independencia a los servicios. Los servicios orquestados no "conocen" (y no necesitan conocer) que están implicados en un proceso de composición y que forman parte de un proceso de negocio de nivel más alto. Solamente el coordinador central de la orquestación es "consciente" de la meta a conseguir, por lo que la orquestación se centraliza mediante definiciones explícitas de las operaciones y del orden en el que se deben invocar los servicios (véase figura 2).

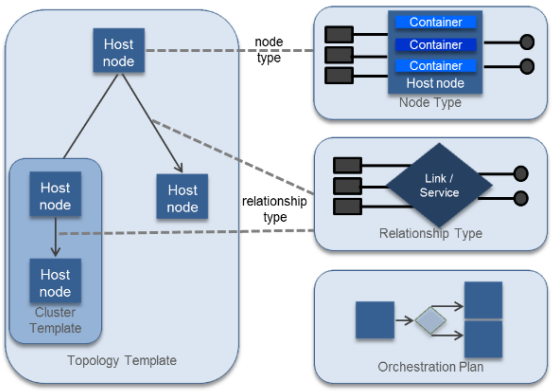


Figura 2: Orquestación en clúster de contenedores

En este trabajo se ha decidido usar orquestación, debido a las siguientes razones:

- Los microservicios no tienen que conocer el proceso de negocio.
- Existe una pieza central a la que se le puede preguntar por el estado de los procesos, estadísticas, etc.
- Permite la migración de versiones de procesos comenzados.

Gracias a la orquestación de contenedores [25] se puede trabajar de forma transparente frente al aumento de carga de trabajo, falla de algún nodo, creación de nuevos contenedores, etc. El orquestador decide cuándo adaptar el sistema (por ejemplo, para iniciar la ejecución de un contenedor) y donde la adaptación tendrá lugar (por ejemplo, en qué nodo se ejecutarán los contenedores).

Orquestar contenedores, aísla al usuario de los detalles de implementación, lo cual provee transparencia en el acceso y ejecución de los contenedores. Este aspecto es importante cuando se desea implementar replicación y distribución de datos masivos con el objetivo de lograr balanceo de carga y auto escalado horizontal, dependiente de la sobrecarga del volumen de información. Además, la orquestación, permitirá un aprovisionamiento de recursos bajo demanda y un escalado horizontal de la aplicación desarrollada.

En resumen, se puede decir que un orquestador de contenedores es la herramienta, que permite crear un clúster de alta disponibilidad de virtualización de contenedores, y lo dota de un sistema de orquestación dinámica de servicios a nivel de múltiples servidores.

Existen varios productos que permiten realizar la orquestación de contenedores, entre los cuales se puede mencionar: Kubernetes [26], Marathon Mesos [27], Conductor [28] y Docker Swarm [29]

4. Herramienta Seleccionada

En este trabajo de investigación se ha decidido trabajar con las tecnologías de contenerización provistas por el ecosistema Docker [30], cuya plataforma se muestra en la figura 3.

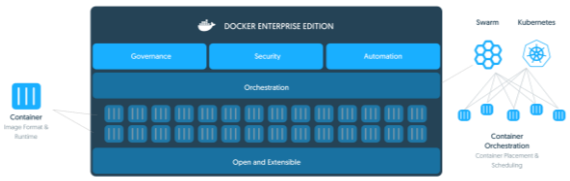


Figura 3: Plataforma Docker

Docker es un proyecto "open source" que permite crear aplicaciones en contenedores de software que son ligeros, portables y autocontenidos.

La implementación de la orquestación se realizó con Docker Swarm para orquestar los contenedores construidos con Docker. De esta manera, es posible convertir un conjunto de nodos (clúster físico) en un nodo único virtual para aprovechar los recursos de cada una de las máquinas.

El reparto de contenedores entre los nodos esta implementado por Docker y para la comunicación con el clúster se utiliza la API estándar de Docker, por lo que cualquier aplicación o herramienta que se comunicaba con un proceso Docker, puede comunicarse con Docker Swarm para escalar a varios nodos.

Este orquestador, posee una arquitectura maestro-esclavo (véase figura 4). Cuando se tienen que distribuir tareas en el swarm (enjambre), los usuarios transfieren los servicios al clúster Manager (CM), que actúa de maestro en el clúster.

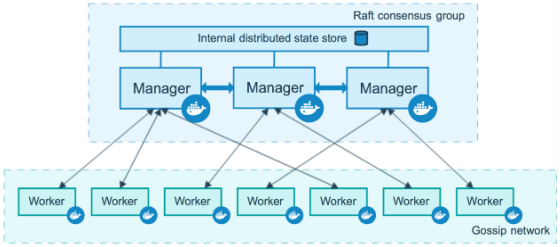


Figura 4: Arquitectura de Docker Swarm

El CM es responsable de la planificación de los contenedores en el clúster y actúa de interfaz primaria a la hora de acceder a recursos de Swarm. La tarea más importante del CM es enviar cada unidad de trabajo (tasks) a los esclavos subordinados (worker nodes).

En cada esclavo funciona un agente el cual, recibe las tareas y entrega al nodo maestro los informes de avance.

Para crear el clúster se debe instalar Docker en todos los hosts y abrir un puerto TCP en cada nodo para la comunicación con el gestor de Swarm. Tras este paso, hay que configurar el contenedor de la imagen Swarm con el rol de agente o gestor.

Una característica muy importante de Swarm es la posibilidad de modificar el número de nodos durante la ejecución gracias a los servicios de descubrimiento.

5. Caso de Estudio

Una de las características más restrictiva de las aplicaciones para trabajar con datos masivos, es la necesidad de procesar ingentes cantidades de datos en un tiempo razonable. Más allá de las técnicas usadas para el procesamiento: paralelas, distribuidas o hibridas, es necesario minimizar los tiempos de acceso a los datos. Para ello, una solución es la replicación, gracias a la cual se puede lograr balanceo de carga, procesamiento concurrente y paralelización

de consultas. Para lograr esto, es necesario contar con una plataforma de hardware que permita orquestar los contenedores con el fin de lograr los objetivos planteados.

En esta investigación se ha trabajado con orquestación de contenedores para el manejo de réplicas en un clúster de nodos virtuales. La idea principal, es realizar la replicación de una aplicación con el objeto de acceder a datos distribuidos en el clúster virtual. De esta manera se podrá acceder desde cualquiera de las réplicas a los datos distribuidos logrando balancear la carga en las consultas y disminuir los tiempos de respuesta.

Para lograr esto, se definió un clúster de máquinas virtuales como primer escenario de trabajo.

La aplicación cuenta con tres pilas de servicios:

- getstartedlab_web (5 réplicas)
- getstartedlab_visualizer (1 replica)
- getstartedlab_redis (1 replica)

Se ha definido una replicación con cardinalidad cinco para el servicio *getstartedlab_web*. Para el acceso a los datos se ha construido una aplicación web escrita en python, el framework flask (incorpora un servidor web) y una base de datos redis. En *getstartedlab_visualizer* se muestran los servicios Docker que se ejecutan en el swarm dentro del cluster, y en *getstartedlab_ redis* se agrega una base de datos redes para almacenar datos de la aplicación.

Toda la configuración específica de cada uno de los contenedores se plasmó en un Dockerfile como lo muestra la figura 5. Con este archivo, se pueden indicar las versiones de la imagen de cada contenedor que el servicio necesita, agregar las librerías faltantes a la imagen, etc.

```
FROM webdevops/php-apache

ENV TZ=America/Argentina/Buenos_Aires

RUN apt-get update \
    && apt-get install php-pgsql -y \
    && echo display_errors=Off >> /opt/docker/etc/php/php.ini \
    && echo short_open_tag=On >> /opt/docker/etc/php/php.ini \
    && echo extension=php_pgsql.so >> /opt/docker/etc/php/php.ini

EXPOSE 80 443
```

Figura 5: Archivo Dockerfile

Para la configuración de los microservicios se usó Docker Compose, el cual es una herramienta para definir y correr aplicaciones que utilicen múltiples contenedores. Las configuraciones de los servicios que la aplicación necesite se especifican en un archivo .yaml. Una característica importanre es que Docker Compose puede trabajar en todos los ambientes ya sea producción, desarrollo o testing. Dado que en este caso de estudio se necesitaba de múltiples servicios, resultó ser de gran ayuda para determinar los puertos de cada microservicio, como así también los parámetros de subred, niveles de replicación, volúmenes espejados, como también la relación de dependencia entre microservicios.

Una vez creados y configurados los contenedores, se crea el clúster de máquinas virtuales, como se muestra a continuación:

```
docker-machine create --driver virtualbox myvm1
docker-machine create --driver virtualbox myvm2
```

Para orquestar los contenedores en los nodos virtuales se selecciona el nodo destinado a ser el manager y se ejecuta el comando que se muestra en la figura 6:

```
$ docker swarm init --advertise-addr 192.168.1.80
Swarm initialized: current node (cf2mywzpa3pd77z23u5jciqwi) is now a manager.

To add a worker to this swarm, run the following command:
    docker swarm join \
    --token SWMTKN-1-50ed1k935u9qgvrs8alhpzf1awgdil2dmfs4zgpd8ue2ltkmww-3y9tb0pjnieqao9ahkutpvpxe \
    192.168.1.80:2377

To add a manager to this swarm, run the following command:
    docker swarm join \
    --token SWMTKN-1-50ed1k935u9qgvrs8alhpzf1awgdil2dmfs4zgpd8ue2ltkmww-06m3vta11n41hjyr1ytycwqvf \
    192.168.1.80:2377
```

Figura 6: Creación del Manager en Swarm

Posteriormente se levantan las réplicas, como se muestra en la figura 7.

```
$ docker service create --replicas 4 --name redis --update-delay 10s --update-parallelism 2 redis:3.0.6
```

Figura 7: Creación de réplicas

De esta manera queda configura un cluster Swarm con un manager y cuatro worker. A partir de esta configuración es posible acceder a las réplicas desde un front end web.

Es importante aclarar que esta tarea de configuración solo define el contenedor de la aplicación, la cual es la que implemente el almacenamiento, procesamiento o visualización de los datos. En tal sentido, con Docker Compose, se puede armar una aplicación con tantos componentes como se necesite y, posteriormente orquestar la replicación en el clúster.

6. CONCLUSIONES

El exponencial aumento en la cantidad de los datos generados ha exigido el desarrollo de soluciones computacionales que permitan afrontar las necesidades y retos que trae consigo el big data, en cuanto a su volumen, variedad de fuentes y velocidad con que se generan. Con este objetivo es necesario construir aplicaciones escalables horizontalmente, con la capacidad de abstraerse de la complejidad de la plataforma subyacente.

Para lograr este objetivo una alternativa en la construcción de aplicaciones para datos masivos, es migrarse a la arquitectura de microservicios implementados con contenedores. El escalado horizontal, se debe cumplir tanto a nivel de la plataforma como de la arquitectura de hardware, por lo que usar una arquitectura distribuida permite crecer

en la medida que la aplicación lo necesite. Es por ello que la conjunción de microservicios, contenedores y arquitecturas distribuidas es una solución para lograr que el almacenamiento, procesamiento y visualización de grandes volúmenes de datos se pueda realizar en forma eficiente, sin degradar la performance de la aplicación.

En función de estas consideraciones, se puede inferir que trabajar en el desarrollo de aplicaciones para big data de la forma descripta en este trabajo, permite lograr escalabilidad sobre todo en el aprovisionamiento de nuevas réplicas de la aplicación. Esto es fundamental sobre todo cuando se trabaja con aplicaciones que demandan restricciones de tiempo pues permite aumentar o disminuir en forma elástica los contenedores mediante la orquestación.

Por otro lado, es importante destacar que el uso de contenedores permite la interoperabilidad entre todos los microservicios, así como la independencia entre las tecnologías internas de cada microservicio y de tecnologías externas de desarrollo. Este aspecto hace posible la construcción de aplicaciones altamente portables.

7. Futuros Trabajos

El presente trabajo es una primera aproximación a la orquestación de servicios. El grupo de investigación continúa trabajando en el área con el objeto de implementar sobre una arquitectura física distribuida la orquestación.

Por otro lado, se están comenzando a trabajar en ambientes cloud con el objeto de implementar soluciones basadas en Hadoop / Spark dentro de contenedores que se ejecutarán en cluster sobre la plataforma virtualizada. En este sentido la principal idea se direcciona en la implementación de algoritmos de machine learning.

Además, se ha comenzado a trabajar con bases de datos NewSQL (Spanner) y se planea usar Docker junto a Kebertnet para ofrecer el servicio de almacenamiento de grandes cantidades de datos, con una mínima latencia en los query.

Referencias

[1] P. Mell and T. Grance, “The NIST Definition of Cloud Computing,” 2011.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Futur. Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[3] S. M. Babu, A. J. Lakshmi, and B. T. Rao, “A study on cloud based Internet of Things: CloudIoT,” in *2015 Global Conference on Communication Technologies (GCCT)*, 2015, pp. 60–65.

[4] A. Cocchia, “Smart and Digital City: A Systematic Literature Review,” Springer, Cham, 2014, pp. 13–43.

[5] A. Katal, M. Wazid, and R. H. Goudar, “Big data: Issues, challenges, tools and Good practices,” in *2013 Sixth International Conference on Contemporary Computing (IC3)*, 2013, pp. 404–409.

[6] M. Parashar, X. Li, and Wiley InterScience (Online service), *Advanced computational infrastructures for parallel and distributed adaptive applications*. John Wiley & Sons, 2010.

[7] H.-M. Chen, R. Kazman, and S. Haziyev, “Agile Big Data Analytics for Web-Based Systems: An Architecture-Centric Approach,” *IEEE Trans. Big Data*, vol. 2, no. 3, pp. 234–248, Sep. 2016.

[8] A. S. Tanenbaum and M. Van Steen, *Distributed Systems: Principles and Paradigms, 2/E*. 2007.

[9] D. B. Kahanwal and D. T. P. Singh, “The Distributed Computing Paradigms: P2P, Grid, Cluster, Cloud, and Jungle,” Nov. 2013.

[10] A. Reuther *et al.*, “Scalable system scheduling for HPC and big data,” *J. Parallel Distrib. Comput.*, vol. 111, pp. 76–92, Jan. 2018.

[11] J. Tu, J. Cheng, and L. Han, “Big Data Computation for Workshop-Based Planning Support,” in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, 2015, pp. 1510–1514.

[12] K. Bakshi, “Microservices-based software architecture and approaches,” in *2017 IEEE Aerospace Conference*, 2017, pp. 1–8.

[13] N. Dragoni *et al.*, “Microservices: Yesterday, Today, and Tomorrow,” in *Present and Ulterior Software Engineering*, Cham: Springer International Publishing, 2017, pp. 195–216.

[14] N. Dragoni, I. Lanese, S. T. Larsen, M. Mazzara, R. Mustafin, and L. Safina, “Microservices: How To Make Your Application Scale,” Springer, Cham, 2018, pp. 95–104.

[15] L. Zhu, L. Bass, and G. Champlin-Scharff, “DevOps and Its Practices,” *IEEE Softw.*, vol. 33, no. 3, pp. 32–34, May 2016.

[16] N. Kratzke, “About Microservices,

Containers and their Underestimated Impact on Network Performance," Sep. 2017.

[17] R. Morabito, J. Kjallman, and M. Komu, "Hypervisors vs. Lightweight Virtualization: A Performance Comparison," in *2015 IEEE International Conference on Cloud Engineering*, 2015, pp. 386–393.

[18] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and Linux containers," in *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2015, pp. 171–172.

[19] P. Sharma, L. Chaufournier, P. Shenoy, and Y. C. Tay, "Containers and Virtual Machines at Scale," in *Proceedings of the 17th International Middleware Conference on - Middleware '16*, 2016, pp. 1–13.

[20] C. Ruiz, E. Jeanvoine, and L. Nussbaum, "Performance Evaluation of Containers for HPC," Springer, Cham, 2015, pp. 813–824.

[21] S. R. Brus, D. Wirasaet, J. J. Westerink, and C. Dawson, "Performance and Scalability Improvements for Discontinuous Galerkin Solutions to Conservation Laws on Unstructured Grids," *J. Sci. Comput.*, vol. 70, no. 1, pp. 210–242, Jan. 2017.

[22] S. Yangui, M. Mohamed, S. Tata, and S. Moalla, "Scalable Service Containers," in *2011 IEEE Third International Conference on Cloud Computing Technology and Science*, 2011, pp. 348–356.

[23] C. Pahl and B. Lee, "Containers and Clusters for Edge Cloud Architectures -- A Technology Review," in *2015 3rd International Conference on Future Internet of Things and Cloud*, 2015, pp. 379–386.

[24] N. Busi, R. Gorrieri, C. Guidi, R. Lucchi, and G. Zavattaro, "Choreography and Orchestration: A Synergic Approach for System Design," Springer, Berlin, Heidelberg, 2005, pp. 228–240.

[25] W. Gerlach, W. Tang, A. Wilke, D. Olson, and F. Meyer, "Container Orchestration for Scientific Workflows," in *2015 IEEE International Conference on Cloud Engineering*, 2015, pp. 377–378.

[26] Google, "Kubernetes Documentation - Kubernetes." [Online]. Available: https://kubernetes.io/docs/home/?path=users&persona=app-developer&level=foundational. [Accessed: 08-Jun-2018].

[27] Apache, "Apache Mesos - Documentation Home." [Online]. Available: http://mesos.apache.org/documentation/latest/. [Accessed: 08-Jun-2018].

[28] Netflix, "GitHub - Netflix/conductor: Conductor is a microservices orchestration engine - https://netflix.github.io/conductor/." [Online]. Available: https://github.com/Netflix/conductor. [Accessed: 08-Jun-2018].

[29] Docker, "Set the orchestrator type for a node | Docker Documentation." [Online]. Available: https://docs.docker.com/ee/ucp/admin/configure/set-orchestrator-type/. [Accessed: 08-Jun-2018].

[30] Docker, "Docker - Build, Ship, and Run Any App, Anywhere." [Online]. Available: https://www.docker.com/. [Accessed: 10-Jun-2018].

Processing Collections of Geo-Referenced Images for Natural Disasters

Fernando Loor[1,2], Veronica Gil-Costa[2] and Mauricio Marin[1]

[1] *Universidad Nacional de San Luis, Argentina.*
{fernandoloor1, gvcosta}@unsl.edu.ar
[2] *Universidad de Santiago de Chile, Chile.*
mauricio.marin@usach.cl

Abstract

After disaster strikes, emergency response teams need to work fast. In this context, crowdsourcing has emerged as a powerful mechanism where volunteers can help to process different tasks such as processing complex images using labeling and classification techniques. In this work we propose to address the problem of how to efficiently process large volumes of georeferenced images using crowdsourcing in the context of high risk such as natural disasters. Research on citizen science and crowdsourcing indicates that volunteers should be able to contribute in a useful way with a limited time to a project, supported by the results of usability studies. We present the design of a platform for real-time processing of georeferenced images. In particular, we focus on the interaction between the crowdsourcing and the volunteers connected to a P2P network.

Keywords: Geo-referenced images, support platforms for natural disaster, P2P network.

1. Introduction

During the last years, Big Data has become a strong focus of global interest, attracting more and more attention from academia, industry, government and other organizations. The growing flow of data, which comes from different types of sensors, messaging systems and social networks, in addition to more traditional measurement and observation systems, has already invaded many aspects of our daily existence.

Large data, including large geo-referenced data, have great potential to benefit many social applications such as climate change, health, surveillance, disaster response, critical infrastructure monitoring, transport, etc. Geo-referenced data describe elements in relation to the geographical space - location, often with location coordinates in a spatial reference system. The term commonly used for systems that use this type of data is known as GIS - Geographic Information System. The geo-tagged contents of the web, voluntary geographic information (VGI), satellite navigation, etc. are traditionally collected through sensors. [5]. The authors in [9] claim that geo-referenced data from social networks is another form of VGI data.

The information that is posted on social networks can be very useful in case of a natural disaster. As direct witnesses of the situation, people share photos, messages and videos about events that get their attention. In an emergency operations center, these data can be collected and integrated into the management process to improve the general understanding of the situation or rescue actions. This type of volunteer participation is known as crowdsourcing. Digital volunteers have helped to collect pertinent information much faster than officials or people in charge of coordination activities in natural disasters could do alone, with huge potential impacts on the responsibilities of officials in the management of the information. Emergency search and rescue teams already use pre-event remote sensing data when planning operations [16].

In general, the increasing volume and variable format of the large georeferenced data collected pose additional challenges in storage, management, processing, analysis, visualization and verification of data quality. The authors in [14] state that the size, variety and update rate of data sets exceed the capacity of spatial computing technologies and spatial databases commonly used to learn, manage and process data. with a reasonable effort.

This paper describes the design of a distributed platform that combines algorithms, processing techniques and associated software tools to efficiently

label large volumes of georeferenced images from different sources (satellite images, captured by unmanned aerial vehicles, originated in social networks), to optimize the work of digital volunteers in situations of natural disasters in order to build status maps in real time. In particular, we focus on the interaction between the crowdsourcing server receiving new images to be analyzed and the users inter-connected in a P2P network. The crowdsourcing server sends tasks to the peer-volunteers which tags the images or just vote for an option from a given option list. The crowdsourcing server gathers all the votes form the peers and evaluate their contributions. If there is an agreement in the voting (e.g. most of the volunteers vote for the same option), the task is finished, and the results are saved in a database for statistic purposes. Otherwise, the server selects volunteers with a high reputation and sends the task again.

The remaining of this paper is organized as follows. Section 2 presents previous works. In Section 3 we present the design of our proposed platform and the simulation model for the interaction between the crowdsourcing server and the volunteers. Section 4 presents experimental results and Section 5 concludes.

2. Previous works

New forms of georeferenced data collection have emerged that have given rise to completely new data sources and data types of a geographical nature. The data acquired by the public -VGI-, and the data from the geo-sensor networks have led to a greater availability of spatial information. Whereas until recently, the authoritarian data sets dominated the topographic domain, these new types of data expand and enrich geographic data in terms of thematic variation and the fact that they are more user-centered. The latter is especially true for VGI compiled by social media [13].

Some authors claim that "80% of the data is geographical in nature" [8]. Much of the data in the world can be georeferenced, which indicates the importance of georeferenced large data management. The georeferenced data describe objects and things in relation to the geographical space - location - often with coordinates of location in a spatial reference system. The term commonly used for systems that use this type of data is known as GIS - Geographic Information System.

The work in [3] presents a study of how to find out the current locations of users by tracking their mobile devices, such as smartphones. The study mentions services of geo-social networks such as Foursquare used to locate friends and to find nearby shops and restaurants, where many users register in several places and reveal their current location. In other words, it proposes to solve queries using information from different individuals through its mobile devices.

The work presented in [2], describes some examples of how the Department of Defense of the United States uses crowdsourcing to give answers to problems of natural disasters. The authors conclude that there is a great benefit in taking advantage of the power of the crowd, "a process that will continue to mature, evolve and define the way we help others today, tomorrow and in the future."

Barrington et al. [1] presents a review of the state of the art on the use of crowdsourcing and analysis of images, particularly high resolution aerial. This work describes the experiences obtained in the cases of the earthquake in Haiti and in 2008 in China. (Lee and Kang, 2015) describe the impact of georeferenced data in different applications such as marketing and propose a three layers platform to index and analyze images. But this proposal does not consider the collaboration of volunteers.

Ofli et al. [10] propose a hybrid scheme based on automatic techniques and crowdsourcing for aerial image processing. In this case, manual annotations are used to train a supervised learning system. However, this work does not describe the platform used and does not consider the interaction with information collected by people who are in the place of the event.

There are some platforms such as Tomnod (http://www.tomnod.com), GeoTag-X [15], and some research works that address the problem of using volunteers for the processing of georeferenced images [1] [18] [17] [11] [4]. In particular, Tomnod of the company DigitalGlobe is using Artificial Intelligence (AI) driven by crowdsourcing to automatically identify the characteristics of interest in satellite and aerial images. Tomnod runs crowdsourcing campaigns, where volunteers support data mapping by validating the results of an image mining algorithm. GeoTag-X is a research project aimed at researching and evaluating collaborative on-line environments and software tools for creative learning.

3. Proposed methodology

In this section, we present our platform design for processing large number of georeferenced images. Our design focused on real-time applications devised for natural disasters which require a low latency and short response time.

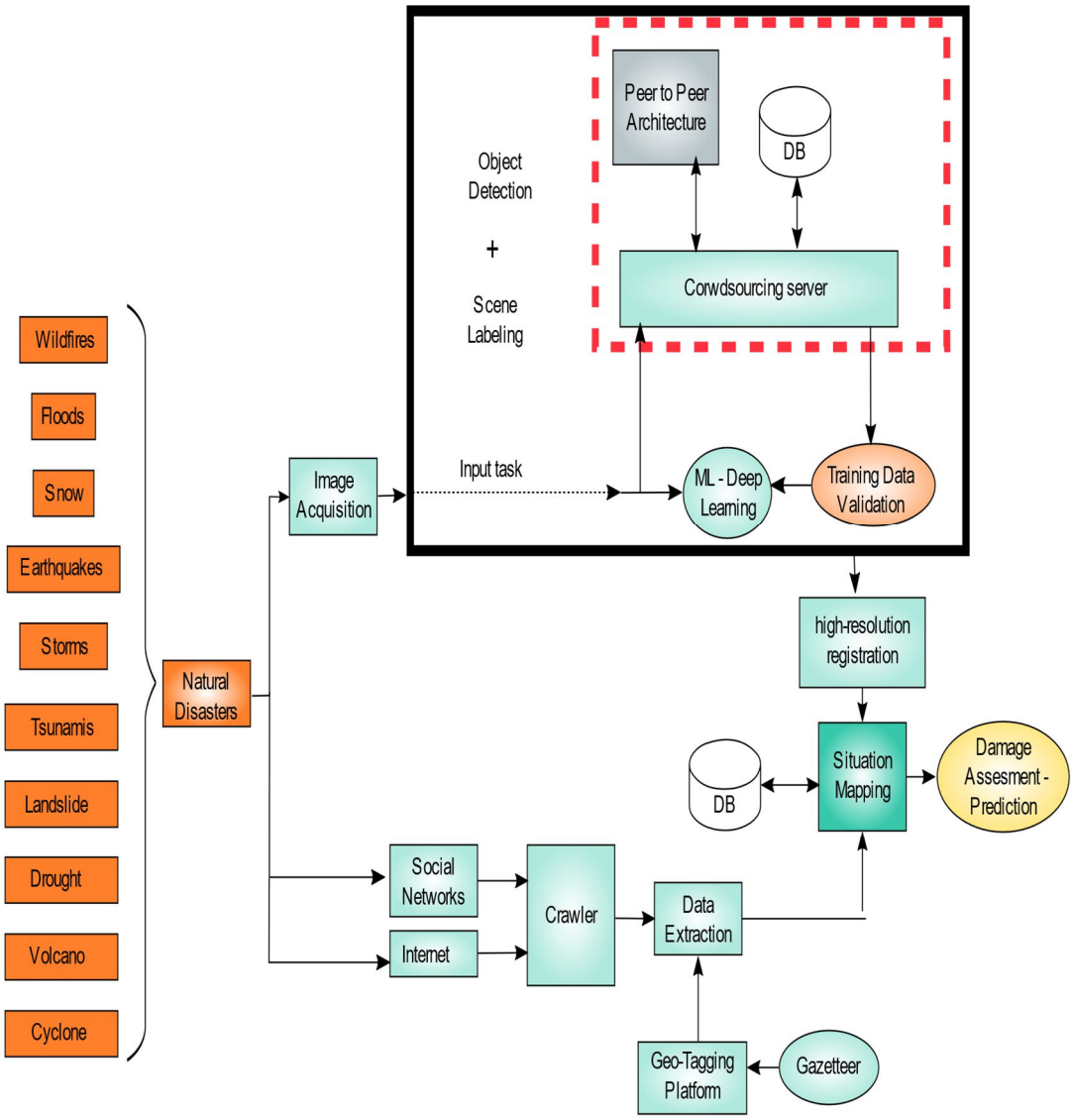


Fig. 1: Proposed distributed platform.

Fig. 1 shows the general components of our proposed platform and how they communicate to each other. Upon a natural disaster, images are captured through different devices such as a drone and smartphones (*Image acquisition*). Then, these images are sent to the *Object detection and scene labeling* component – the black box at the top of the figure. For each incoming image we create a task which will be sent to the crowdsourcing server or to the Machine Learning server. The task contains metadata associated with the image like the GPS coordinates, the image identifier, and a list of options to vote or to label the image. If the machine learning server is not trained for the region where the images come from, the task is sent to the crowdsourcing server. This server will request to different volunteers to help tagging the images. It allows different users to classify images manually, in order to generate new classifiers in real time. Classifiers can detect emerging needs at the time of a disaster.

The results obtained from the crowdsourcing server are used to train the machine learning server. Once this last server is trained, it can start receiving incoming tasks. In this way, we can reduce the amount of work sent to volunteers.

The results achieved by the object detection and scene labeling component are sent to the high-resolution registration component. This last, establishes a correspondence between the image been analyzed and the geographical coordinates to create a situation map. The information inside the situation map is stored in a NoSQL database like MongoDB and can be later use for emergency management and

help for victims.

At the bottom of the figure, we show the components deployed in our proposed platform for text messages crawled from Tweeter or others social networks. Relevant data is extracted from these texts The Geo-Tagging Platform uses the Gazetteer which is a geographic dictionary to parse the tweets to identify which coordinates they belong to.

3.1 Modelling the crowdsourcing server

Fig.2 shows the general scheme of our model for the crowdsourcing server and the P2P network. Our model considers a group of users who execute the tasks delivered by the crowdsourcing server. Each task has information related to the incoming image like the time to live (TTL) of the task, the priority, the image identifier, and the list of options. If the priority feature is on, the priority task is high and should be process immediately.

Peers build an overlay network managed by the Internet service providers (ISP). In particular, our model follows a P2P Distributed Hash Table - DHT [12].

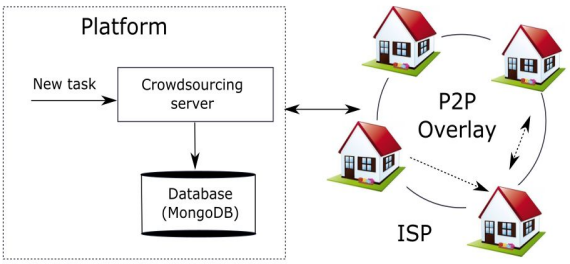


Fig. 2 Simulated scheme.

Internet service providers (ISPs) are responsible for delivering Internet access to clients from a given geographic area. To communicate with other ISPs, it is necessary to access the Internet backbone. The backbone is a shared network which enables communication among ISPs of the world. To make use of the network, ISPs must respect a Service Level Agreement (SLA) contract in which they commit to regulate their traffic to not compromise another ISPs communication.

Each peer holds a task queue which stores tasks with the priority feature given by the TTL. These tasks should be quickly process in order of arrival.

In general, the sequence of steps executed by our model is as follows:

(1) The crowdsourcing server receives incoming tasks from the platform.
(2) A peer becomes visible to the server indicating that he wants to be a volunteer.
(3) The server sends a list of tasks to the peer.
(4) The peer agrees to start processing the tasks and request the first image.
(5) The server sends the first image to the peer and set a time-to-live (TTL).
(6) The peer executes the task and sends the result to the server.
(7) If a total of **H** answers were received for a task, the server evaluates if there is an agreement in the voting.
    a. If so, the task is finished, and the data is sent to a data base server like MongoDB or Casandra.
    b. Otherwise, the server selects new peers with high reputation to send the task again.
(8) If the peer wants to continue collaborating, return to step 4 for the next images.

To reduce the communication between the crowdsourcing server and the P2P network, each peer has an LRU cache memory with images received from the server. Thus, the next time a peer requests an image form the same task list, the peer will search for that image inside the P2P network using the DHT before sending a request to the server.

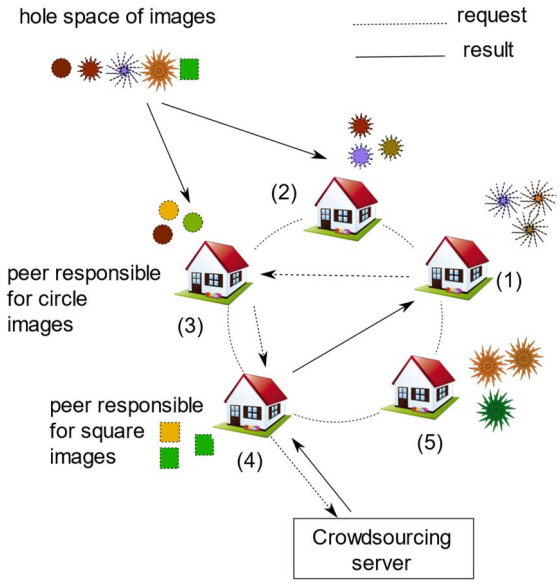


Fig. 3 Communication scheme.

More precisely, as the hole space of images is divided among the peers, each peer is responsible for a particular set of images. As shown in Fig. 3, each peer is responsible for a different set of images (square, circle and the three types of stars). In this example, the peer 1 request a square image of color blue. Then it sends a message to peer 3 which is selected by the DHT. Due to peer 3 does not have the requested image, it sends the message to the next hop in the P2P network. In this case, the message reaches to peer 4, which is the peer responsible for this kind

of images. As peer 4 does not have the blue square image, it sends a request to the crowdsourcing server. The server sends the requested image to the peer 4, which is the responsible. Peer 4 insert this new image into its LRU cache memory. Finally, peer 4 sends the image to the requesting peer 1.

If the TTL of a given task expires, the server sends the results achieved at the moment to the database server. Additionally, some volunteers can offer to continue helping with the voting/tagging tasks. In this case, the value of **H** – number of expected answers per task- can be increased. This helps to improve the quality of the results and get a more adequate consensus of the tasks of voting/tagging.

The server applies two policies for the TTL of each task. (1) If the TTL of a given task expires and there are at least **H** results received from the peers, the server ranks the results. (2) Otherwise, if the TTL expires and there are less than **H** results, the server extends the TTL and sends additional request with high priority (the priority featured turned-on) to frequently active volunteers.

4. Experimental results

In this work, we evaluate the performance of the crowdsourcing server and its interaction with the network of volunteer forming a P2P network. We have built a simulator that implements a transport layer, a P2P overlay and our caching proposal. Pastry [12] has been used as the overlay network in our experimentation. We have also simulated the Web crowdsourcing server, a generator which creates the incoming tasks and a database server which receives data form the crowdsourcing server. The simulation is divided in simulation time windows of 100 units of time.

4.1. Simulation approach

The simulation model of this work uses a processes and resources approach. Processes represent the crowdsourcing server, and the peers in charge of transaction processing. Resources are artifacts such as the data of the incoming messages, global variables like the input queue of each process and also the CPU and the communication network. The simulation program is implemented using LibCppSim [7], where each process is implemented by a co-routine that can be blocked and unblocked at will during simulation.

The operations *hold*(), *passivate*() and *activate*() are used for this purpose. Thus, a coroutine C_i can be paused for a given amount of time Δ_t -which represents the duration a task. Once the simulation time Δ_t has expired, the coroutine C_i activates itself if a hold() operation was previously executed. Otherwise, the coroutine C_i is activated by another coroutine C_j using the activate() operation. This last case allows to represent the interaction among the different components of the simulated platform.

The simulated architecture assumes a classical DHT overlay composed by **N** physical nodes (or peers) and **K** object-keys (task with images ids) mapped onto a ring. Objects (images) stored in a DHT such as Pastry [12], have a responsible peer in the network that is the peer with the closest ID to the key of the object. Thus, any given peer is responsible for a fraction of these images and (on request) it must contact the crowdsourcing server to get them.

4.2. Experiment Settings

We evaluate different metrics and costs between the crowdsourcing server and the peers by setting the parameters of our simulator as shown in Table 1.

Table 1 Parameters setting for the simulation.

Parameter	***Value***
Network size (num. peers)	{100,500,1000, 1500}
Cache size in each peer	{50,100,150}
Total answers expected by the crowdsourcing server (**H**)	{10,15,20,25}
Arrival rate	{500, 1000, 2000}

4.3. Gini Coefficient

To measure load balance among peers we use a metric based on Lorenz curves called the Gini coefficient, which is a metric commonly used on other fields like economics and ecology.

If all peers have the same load, the Lorenz curve is a straight diagonal line, called the line of equality or the perfect load balancing. If there is any imbalance, then the Lorenz curve falls below the line of uniformity. The total amount of load imbalance can be summarized by the Gini coefficient G, which is defined as the relative mean difference, i.e. the mean of the difference between every possible pair of peers, divided by their mean load. G values range from 1 to 0. The value 0 is achieved when all peers have the same load. The value 1 is achieved when one peer receives the whole system load while the remaining peers receive none. Therefore, when G approaches 0, global load imbalance is small, and when G approaches 1 the imbalance is large.

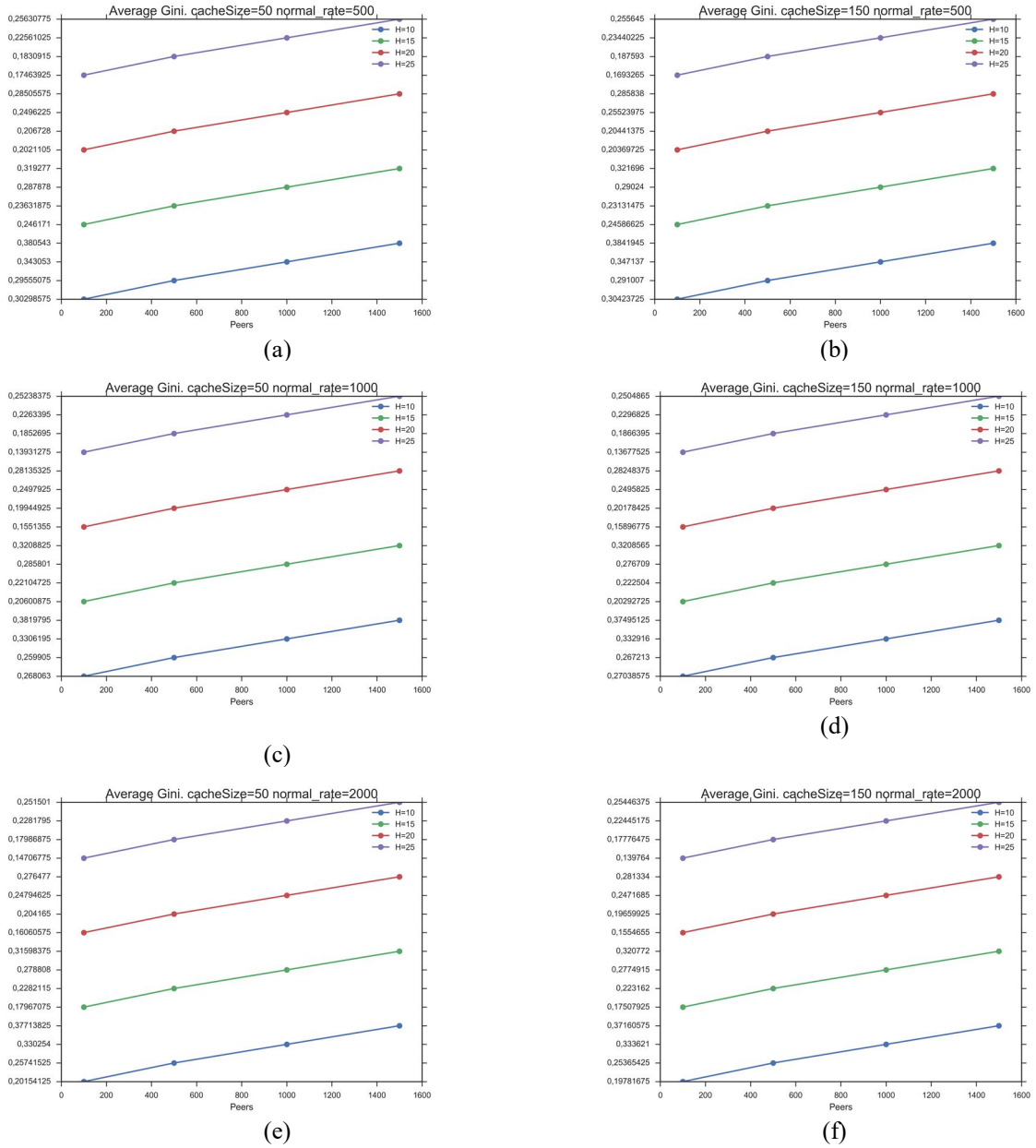


Fig. 4: Gini coefficient for different cache size, number of peers, arrival rate and different values of H.

In Fig. 4, the y-axis shows the Gini coefficient for different configurations. The x-axis shows the values of H. In Fig. 4 (a)(b)(c)(d)(e) and Fig. 4(f) we vary the arrival rate from 500 to 2000 requests per seconds and the cache size from 50 to 150. Results obtained with a cache size of 100 are very similar to the ones presented in these figures.

In all cases, the Gini coefficient is lower than 50%, meaning that the workload tends to be balance among the peers. Moreover, as we increase the value of H (in the x-axis) the imbalance tends to be lower, as more peers execute similar amounts of tasks.

On the other hand, the arrival rate and the cache size have very low impact on the performance achieved by the peers. However, as we increase the number of peers, the workload tends to be more balanced. That's because, the tasks are evenly distributed among the peers and the query arrival rate is high tending to produce different queue size inside each peer.

4.4. Number of Hops

Fig. 5 shows the average number of hops that a massage has to go thought, inside the P2P network, before reaching the requested image. The x-axis shows the number of peers selected by the crowdsourcing server to send the tasks. We show the results obtained for a cache size of 150 and an arrival rate of 2000 tasks per second. Results obtained with other parameters are very similar, because the number

of hops does not depend on the cache size of the peers, neither on the arrival rate.

As expected, the number of hops tends to increase with a larger number of peers in the network. On the other hand, the H value impacts on the number of hops. A larger H value tends to reduce the number of hops required to find the image inside the P2P network. That's because, as more peers are involved in the process to solve a task (e.g. voting or tagging), the images for those tasks are going to be used by more peers, and therefore a new peer requesting an image would probably find that image in a near by peer.

With few peers, the H value does not impact on the number of hops. That's because the network size is small and the peer for a give image is found more quickly, without having to visit other peers along the way.

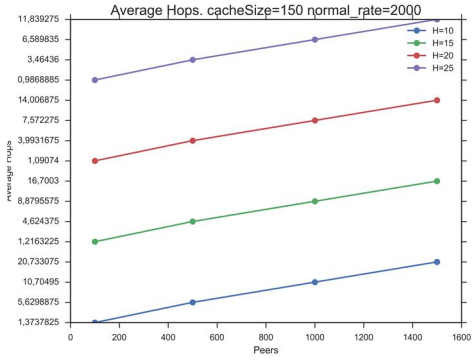


Fig. 5: Average number of hops acieved by different parameters of our simulated platform.

4.5. Latency

In Fig. 6, we show the average latency reported by sending messages inside the P2P network. We show results for different arrival rates and a cache size of 150. Notice that the cache size does not impact on the latency.

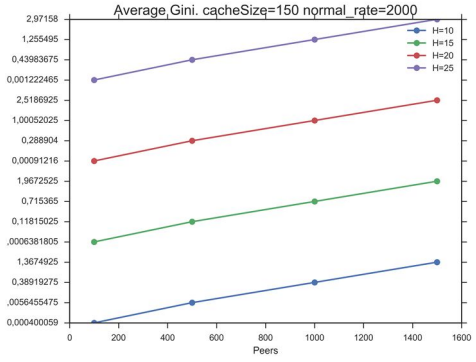
As expected, more peers in the network tends to increase the latency, due to more hops are visited before reaching the final destination. That is, the peer holding the requested image.

Also, the arrival rate tends to smoothly increase the latency in the P2P network, due to more messages are present in the network at the same time and those messages compete for the network resource.

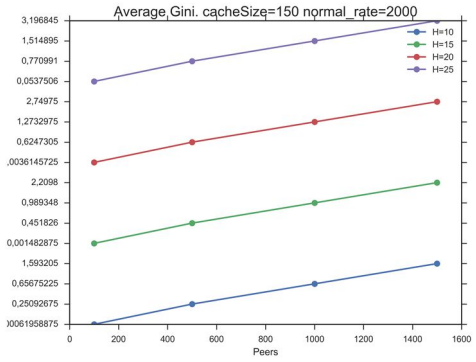
Finally, with a larger H value, the latency also increases because more peers have to request the same tasks and images to process. These requests are messages traversing the P2P network.

5. Conclusions

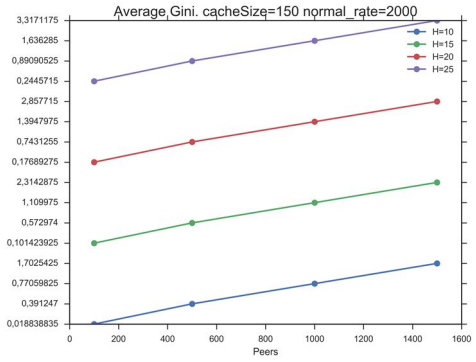
In this work, we presented a new platform to support decision making in cases of natural disasters, through the processing of information such as georeferenced images in real time. We presented the components of our proposal and how they communicate to each other.



(a)



(b)



(c)

Fig. 6: Latency reported by the P2P network for different arrival rates (a) 500, (b) 1000 and (c)

We also modelled the interaction between the crowdsourcing server and the volunteers forming a P2P network. We simulated this model to analyze possible bottlenecks and the benefits of using cache memories in the peers to avoid accessing the crowdsourcing server for each new requested task.

We evaluated different metrics to analyze the effect of communication and the number of peers volunteers on the performance of the platform.

Results show that the number of selected peers as volunteers affects the latency of the communication in the P2P network as the number of peers increases.

Acknowledgements

This research was partially funded by PICT 2014 N° 2014-01146.

Competing interests

The authors have declared that no competing interests exist.

References

[1] L. Barrington, G. Shubharoop G. Marjorie, H.N. Shay, J. Berger, S. Gill, A. Yu-Min Lin and C. Huyck. "Crowdsourcing earthquake damage assessment using remote sensing imagery". Annals of Geophysics. vol. 54, no. 6. ,2011

[2] D. Becker and S. Bendett. "Crowdsourcing Solutions For Disaster Response: Examples And Lessons For The US Government, Humanitarian Technology: Science, Systems and Global Impact", in HumTech, 2015.

[3] M. Choy, J.-G. Lee, G. Gweon, D. Kim. "Glaucus: exploiting the wisdom of crowds for location-based queries in mobile environments", in Proceedings of 8th In-ternational AAAI Conference on Weblogs and Social Media, pp.61–70, 2014.

[4] P. Díaz, J. M. Carroll and I. Aedo. "Coproduction as an Approach to Technology-Mediated Citizen Participation in Emergency Management", in Journal of Future Internet, vol. 3, no. 3, 2016.

[5] M. R. Evans, D. Oliver, X. Zhou and S. Shekhar. "Spatial Big Data: Case Studies on Volume, Velocity, and Variety". in H. A. Karimi (Ed.), Big Data: Techniques and Technologies in Geoinformatics, pp. 149-176, 2014

[6] L. Jae-Gil and K. Minseo. "Geospatial Big Data: Challenges and Opportunities", in Big Data Research, pp. 74–81, 2015.

[7] M. Marzolla. "Libcppsim: a Simula-like, portable process-oriented simulation library in C++", in ESM, pp. 222–227, 2004.

[8] C. D. Morais. "Where is the Phrase "80% of Data is Geographic" From?". from http://www.gislounge.com/80-percent-data-is-geographic/, 2012.

[9] S. Newsam. "Crowdsourcing What Is Where: Community-Contributed Photos as Volunteered Geographic Information", in Knowledge Discovery from Community- Contributed Multimedia, pp 36-45, 2010.

[10] F. Ofli, P. Meier, M. Imran, C. Castillo, D. Tuia, N. Rey, J. Briant, P. Millet, F. Reinhard, M. Parkan, and S. Joost. "Combining human Computing and Machine Learning to Make Sense of Big (Aerial) Data for Disaster Response", in Journal of Big Data, vol. 00, no. 00, pp 1-13, 2016.

[11] T. Onorati, P. Díaz. "Giving meaning to tweets in emergency situations: a semantic approach for filtering and visualizing social data", in Journal of SpringerPlus, vol. 5, no. 1, 2016.

[12] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in Middleware, ser. LNCS, vol. 2218, pp. 329–350, 2001.

[13] M. Sester, J.J., Arsanjani, R.., Klammer, D. Burghardt and J-H. Haunert. "Integrating and Generalising Volunteered Geographic Information Abstracting Geographic Information in a Data Rich World" in *Methodologies and Applications of Map Generalisation*, pp. 119-155, 2014.

[14] S. Shekhar, V., Gunturi, M.R., Evans, and K. Yang. "Spatial Big-Data Challenges Intersecting Mobility and Cloud Computing", in Proceedings of the 11th ACM International Workshop on Data Engineering for Wireless and Mobile Access – MobiDE, 2012.

[15] C. Smith. "A Case Study of Crowdsourcing Imagery Coding in Natural Disasters", in Data Analytics in Digital Humanities, pp. 217-230, 2017.

[16] S. Thorvaldsdóttir, E. Birgisson and R. Sigbjornsson. "Interactive on-site and remote damage assessment for urban search and rescue", Journal of Earthq. Spectra, vol. 27 no. (S1), pp S239-S250, 2011.

[17] C. Turk. "Cartographica incognita: 'Dijital Jedis', Satellite Salvation and the Mysteries of the 'Missing Maps'", Journal of The Cartographic, pp. 14-23, 2016.

[18] N. Witjes, P. Olbrich and I. Rebasso. "Big Data from Outer Space: Opportunities and Challenges for Crisis Response", in Yearbook on Space Policy 2015: Access to Space and the Evolution of Space Activities, pp 215-225, 2017.

Secure Computer Network: Strategies and Challengers in Big Data Era

Mercedes Barrionuevo[1], Mariela Lopresti[1], Natalia Miranda[1], and Fabiana Piccoli[1]

[1]*LIDIC, Universidad Nacional de San Luis, San Luis, Argentina*
{mdbarrio, omlopres, ncmiran, mpiccoli}@unsl.edu.ar

Abstract

As computer networks have transformed in essential tools, their security has become a crucial problem for computer systems. Detecting unusual values from large volumes of information produced by network traffic has acquired huge interest in the network security area. Anomaly detection is a starting point to prevent attacks, therefore it is important for all computer systems in a network have a system of detecting anomalous events in a time near their occurrence. Detecting these events can lead network administrators to identify system failures, take preventive actions and avoid a massive damage.

This work presents, first, how identify network traffic anomalies through applying parallel computing techniques and Graphical Processing Units in two algorithms, one of them a supervised classification algorithm and the other based in traffic image processing. Finally, it is proposed as a challenge to resolve the anomalies detection using an unsupervised algorithm as Deep Learning.

Keywords: Computer Network, Network Security, Anomalies and Attacks, Big Data, High Performance Computing, Machine Learning.

1 Introduction

In the last decade, World Wide Web (WWW) together its fast and growing development, has produced changes in information technologies. Although, it has brought great benefits in many areas, it also has some drawbacks.

Even though computer networks provide global connection and access for all information type, also provides malicious users with new tools for their destructive purposes. The costs of temporary or permanent damage caused by unauthorized access to computer systems have prompted organizations to implement complex systems to control the flow of data on their networks.

Threats to a data network are made by a packages set with specific characteristics to detect system vulnerabilities. These hazards represent risks for any organization and can be used to carry out attacks. Detecting possible attacks requires to count with several methods and strategies to classify traffic. This area is a wide interest problem, especially in emerging areas such as big or massive data.

Big Data is not a technology in itself, it is a work approach to obtain information (value) and benefits from large volumes of data. For this, it necessary take into account:

- How to capture, manage and take advantage of data.
- How to secure data and its derivatives, as well as its validity and reliability.
- How to share data to obtain improvements and benefits in the organization.
- How to communicate data (visualization techniques, formats and tools) to facilitate decision making and subsequent analysis.

A Big Data problem is defined through the 7 Vs: Volume, Velocity, Variety, Veracity, Validity, Visualization and Value. Each of these characteristics is briefly detailed in section 2.2.

For each of above aspects, an interdisciplinary work is necessary that includes areas of High Performance Computing (HPC), Image Processing and Machine Learning. The problem of traffic analysis and detection of possible network attacks fit to this kind of work and, in consequence, they can be considered a Big Data problem.

The use of data-intensive applications carry on users to consider new HPC configurations to work with huge amounts of data. In effect, the International Data Corporation (IDC) has incorporated the term High Performance Data Analytics (HPDA) [1] to represent Big Data analysis over HPC configurations. HPDA provides the ability to solve a new class of computational problems of data-intensive.

The objective of this work is to develop models for search of anomalies in the network traffic and detect attacks to data networks. These models search to identify patterns that deviate from normal behavior. For this, it is proposed to analyze the network traffic by

means of different techniques that allow to obtain concrete results and a near time to detect an attack and to make quick decisions.

This paper is organized as follows: the next section describes theoretical concepts need to this work. Section 3 explains the relation between analysis of network traffic and big data problems. In section 4, the main characteristics of proposed model are detailed, and in Section 5, some experimental results of performance are sketched. Finally, the conclusion and future works are drawn.

2 Background

This work involves many concepts, among them we emphasize computers networks traffic and its anomalies, extraction methods of packages characteristics, Machine Learning algorithms and Big Data. In this section, we describe each one of them.

2.1 Computer Networks Data Traffic

Network traffic provides information about what travels by network. The most common data types are log data, such as Internet Protocol (TCP/IP) records, event logs, internet access data, Network Management Protocol (SNMP) data reporting, among others [2]. This information is necessary for network security, specifically for anomalous events detection. Fig. 1 illustrates an example of TCP/IP traffic, the rows detail individual network traffic and the columns are specific characteristics of each traffic. In the example, the first column is a session index for each connection, and the second says when the connection has occurred [2].

```
1  06/24/1998 08:12:58 00:00:01 ntp/u 123 123 172.016.112.020 192.168.001.010 0 -
2  06/24/1998 08:12:58 00:00:01 ntp/u 123 123 172.016.112.020 192.168.001.010 0 -
3  06/24/1998 08:15:52 00:00:04 smtp 1024 25 172.016.114.169 195.115.219.108 0 -
4  06/24/1998 08:15:55 00:00:01 domain/u 53 53 192.168.001.010 172.016.112.020 0 -
5  06/24/1998 08:15:55 00:00:01 domain/u 53 53 192.168.001.010 172.016.112.020 0 -
6  06/24/1998 08:15:55 00:00:02 smtp 1025 25 172.016.114.169 196.227.033.189 0 -
7  06/24/1998 08:17:08 00:00:04 smtp 1026 25 172.016.113.084 195.115.218.108 0 -
8  06/24/1998 08:17:11 00:00:02 smtp 1027 25 172.016.113.084 196.227.033.189 0 -
9  06/24/1998 08:17:19 00:00:02 smtp 1028 25 172.016.112.149 195.115.219.108 0 -
10  06/24/1998 08:17:36 00:00:01 domain/u 53 53 192.168.001.010 192.168.001.020 0 -
11  06/24/1998 08:17:36 00:00:01 domain/u 53 53 192.168.001.010 192.168.001.020 0 -
12  06/24/1998 08:17:37 00:00:02 smtp 1029 25 172.016.114.169 194.027.251.021 0 -
13  06/24/1998 08:17:38 00:00:02 smtp 1040 25 172.016.114.169 194.007.248.153 0 -
14  06/24/1998 08:17:39 00:00:02 smtp 1049 25 172.016.114.169 197.182.091.233 0 -
15  06/24/1998 08:17:40 00:00:02 smtp 1051 25 172.016.114.169 195.115.218.108 0 -
16  06/24/1998 08:17:41 00:00:02 smtp 1052 25 172.016.114.169 196.227.033.189 0 -
17  06/24/1998 08:17:45 00:00:01 smtp 1104 25 172.016.114.169 135.008.060.182 0 -
19  06/24/1998 08:18:07 00:00:01 eco/i - - 192.168.001.005 192.168.001.001 0 -
20  06/24/1998 08:18:07 00:00:01 eco/i - - 192.168.001.005 192.168.001.001 0 -
21  06/24/1998 08:18:07 00:00:01 ecr/i - - 192.168.001.001 192.168.001.005 0 -
22  06/24/1998 08:18:20 00:00:04 smtp 1107 25 172.016.114.207 196.227.033.189 0 -
23  06/24/1998 08:18:29 00:00:04 http 1108 80 172.016.113.204 205.181.112.065 0 -
24  06/24/1998 08:18:29 00:00:04 smtp 1109 25 172.016.112.194 196.037.075.159 0 -
25  06/24/1998 08:18:32 00:00:02 smtp 1110 25 172.016.112.194 196.227.033.189 0 -
26  06/24/1998 08:18:33 00:00:01 http 1111 80 172.016.113.204 205.181.112.065 0 -
27  06/24/1998 08:18:33 00:00:01 http 1112 80 172.016.113.204 205.181.112.065 0 -
28  06/24/1998 08:18:33 00:00:01 http 1113 80 172.016.113.204 205.181.112.065 0 -
29  06/24/1998 08:18:33 00:00:01 http 1115 80 172.016.113.204 205.181.112.065 0 -
30  06/24/1998 08:18:33 00:00:01 http 1116 80 172.016.113.204 205.181.112.065 0 -
```

Figure 1: Example of TCP/IP Traffic.

Data traveling on network can provide important information about user and system behaviors. These data can be collected with some commercial products or specific software, for example TCP/IP data can be captured using different tools, called sniffers. Network traffic is composed of packets, flows and sessions. A packet is a data unit exchanged between a source and a destination on the Internet or another TCP/IP-based network; a network flow is a one-way packets sequence between two endpoints; and the session data represents communication between computers. A communication involves the interchange of multiple flows. Traditionally, an IP flow or tuple contains a set of attributes, for this work, the more important are: source and destination IP address, source and destination port, and protocol type. The protocol type, if you consider the 4-layer TCP/IP model, can be TCP or UDP (layer 3) or ICMP (layer 2). This information allows to establish a behavior baseline or normal pattern of network traffic, and in consequence to identify unexpected or unwanted conduct, called anomalous traffic. Therefore, an analysis strategy by anomalies bases on the traffic description in normal conditions to classifies as anomaly all patterns that move away from it. In order to obtain this dataset, there are several techniques, some of which are mentioned in the following subsection.

2.1.1 Analysis of a Network Packet

Studying the particular aspects of network traffic, it is necessary to extract only information from data packets and then process them. There are different techniques of extraction and processing, some of them are:

- Graphical representation of raw data: Generally, the representations are 2D and 3D scatter graphics, time-based graphics, histograms, pie charts, or diagrams.
- Statistical information and pattern extraction: They are based on average calculations, time distributions and probability distribution functions.
- Analysis based in rule (signatures), anomaly detection and policy: All traffic inspection analyzer that look for coincidences with a particular rule or signature belong this category. Rules are defined as values for certain fields in the header or a combination of several of them. These techniques are used in intrusion detection systems (IDS), such as Snort [1].
- Flow-based analysis: It focus on network traffic management as flow. The most of network information exchanged is oriented to connection (and non-oriented to packet), the analysis can take advantage of this. A clear example of typical network flow is a TCP connection, where the data exchanged are governed by the TCP state machine [3].

Each of these techniques is suitable for specific situations, also it is possible combine them. This work is based on flow analysis and rule-based analysis.

[1] www.snort.org

2.1.2 Usual Attacks

One of the biggest challenges for network administrators is to detect attacks on computer networks. An attack implies to take advantage of a computer system vulnerability (operating system, application software, or user's system) for unknown purposes but, usually, causing damage. Therefore, it is impossible to make a complete classification of all the actual attacks and possible weaknesses of the networks, even more when networks are connected to Internet. The denial of service (DoS) and distributed denial of service (DDos) attacks are of great interest today. A DoS attack comes from a single entity and its goal is to turn out unavailable the resources or services of a computer. There are different types, in particular this work has focused on the DoS attacks: Smurf, Fraggle and Land [4] [5], each one of them has the following characteristics:

- Smurf: This attack uses ICMP protocol to send a broadcast ping with a false source address. There are different ways to do a ping, they are:
  - Normal Ping: One or more ICMP echo requests are sent to a system, which responds with one or more ICMP echo replies. Thus, this operation verifies remote system.
  - Broadcast Ping: This ping sends an ICMP echo request to a broadcast address. Each system responds to sender, flooding it with ICMP echo replies.
  - Broadcast ping with false source: A broadcast ping is sent with victim's source address. Each system in network replies and floods the victim with answers. This operation is a combination of the two previous Ping.

  The pattern to recognize this attack type is to analyze to ICMP protocol, if source and destination IP addresses belong to the same network, and the destination address is a broadcast message.

- Fraggle: When you want to check if a system is working, you can use UDP-based tools instead of ICMP to inspect whether the system is listening by a specific port or not. This is commonly done with different types of vulnerability scans, that are used by attackers or security administrators. For example, if a system listens over Port 19 (TCP or UDP), when a connection is established, the system would respond with a constant character flow. Typically, the source system uses the TCP or UDP Port 7. When the source system begins to receive characters, it knows that the target system is operational and closes the connection. In a Fraggle attack, a broadcast packet is sent with a false address to the victim's port 19, if it has its port 19 open, it answers a constant flow of characters to victim. The pattern is similar to that of Smurf but in this case the protocol is UDP.

- Land: It's an attack using the TCP protocol. It creates an "infinite loop" which is caused by sending a SYN request with the same source and destination IP address. The victim computer responds to itself until a blocking state appears and it does not accept any new requests. Besides, as all processor resources are exhausted, the denial of service happens. To recognize this type of attack, it is necessary to analyze coincidence between the source and destination IP addresses, as well as the same ports.

As mentioned above, there are many other denial of service attacks, but the detection of patterns in them requires a deeper and detailed analysis that escapes this work.

2.2 Big Data

Big Data is defined as a large data set without an implicit organization and a particular structure. The large volume, velocity and variety of data make traditional databases be inadequate to store and quickly retrieve them, their processing capabilities are exceeded. At beginning of the big data era, the main generators of big data were scientific and physical applications, military experiments, simulations, NASA and super computers. Nowadays, there are many sources that contribute to a constant generation of big data, for examples airlines traffic, medical systems of assistance and insurance, stock exchanges, mechanisms and systems of electronic money, mobile applications. In 2012, approximately 500PB of medical care data were generated, 25.000PB will be expected for 2020. According to Stephen Gold, vice-president of Watson (IBM), 90% of all data was created in 2 years ago, 2.5PB are generated per day [6]. He compared Big Data with oil: if it is in the ground, it does not have much value, but it is different when you use it. In consequence Big Data becomes very interesting if you find ways to process and analyze data. There are studies that show how Big Data has contributes to several scopes, such as industries, markets, health, labor market, stock market, retail, real estate, education, finance, environmental research, genomics, sustainability, politics and biological research [7, 8].

Google has become the leader company by its search engine. BigTable, Hadoop and MapReduce are applications of real-world Big Data problems that have revolutionized industry and companies by providing great solutions [7]. Their parallel and distribution computing model provides them the ability to perform complex operations on very large data sets.

By all above said, large data volumes are produced constantly by a lot of applications. Therefore, it is important to understand and discuss all Big Data's V

in order to get information from these large data sets. At present, 7 Vs are considered to Big Data, they are:

1. ***Volume:***: The first V refers to size of data set, it can be measured in TB, PB or more. These data are generated from different sources as social networks, research studies, medical data, spatial images, crime reports, forecasts meteorological, natural disasters, applications, among other.

2. ***Velocity:*** Data are generated at high speed. Smart-phones or the World Wide Web (WWW) contribute with this.

3. ***Variety:*** A data set can contain structured, unstructured data or both. Besides , these data can be different types, for example: records, audios, videos, texts, images, etc. This feature implies a true complexity to work with them.

4. ***Veracity:*** The credibility is given by the accuracy and quality of data set. How sure are you of these data? Or How many data are really of a given type? Taking account that many unstructured data can come from Facebook posts, tweets, LinkedIn posts, etc. Do you trust what you see? This V concentrates the greatest preoccupation, because the objective is to process and analyze data to get good results.

5. ***Validity:*** In this case, the data accuracy is considered respect to intended use. In consequence, the same data set may be valid to one problem and not valid for another.

6. ***Visualization:*** It refers to way that data are presented or showed. Once data are processed, you need to represent them visually to be legible and accessible, beside to find patterns and hidden keys in relation of considered issue.

7. ***Value:*** The value is obtained when data are transformed into information, the information becomes knowledge and this last concludes in a decision.

In all cases, it is necessary considered that results from big data should not exceed the cost of its processing, administration and/or storage.

2.3 Machine Learning

Machine learning is the study of algorithms which can learn complex relationships or patterns from empirical data, and make accurate decisions. Machine learning includes broad range of techniques such as data pre-processing, feature selection, classification, regression, association rules, and visualization [9]. In big data analysis, machine learning techniques can help to extract information from large data sets, identifying hidden relationships.

Machine learning algorithms are classified into two groups: supervised and unsupervised learning. In the first kind, all data is labeled and the algorithms learn to predict output from input data. In second class, data are unlabeled and the algorithms learn to inherent structure from the input data.

The next sub-sections present two algorithms of machine learning, one of each class. Both can be applied in anomalies detection.

2.3.1 Supervised Classification: *k*-NN Algorithm

The classification process builds models capable of determining if an object, from its characteristics, is member or not of a category. A classification is supervised when, in advance, there is an already classified observations set, and it is known to which set belongs each observation. The algorithms dedicated to solve supervised classification problems usually operate with information provided by a set of samples, patterns, examples or training prototypes, all of them are representatives of each classes [10].

In particular, this paper uses a supervised classification algorithm based on neighborhood criteria. It is known as K nearest neighbor (k-NN). The k-NN method and its variants are based this intuitive idea: "Similar objects belong to the same class". The class is determined by the most similar objects. The similarity idea is formally reflected in distance concept, usually the Euclidean distance is used [11]. The calculation of the k-NN can be solved using parallel techniques, in [11] has been shown that the parallel implementation using GPU [12] obtains very good response times.

2.3.2 Neural Networks

A neural network (NN) consists of simple processing units: neurons, and their connections: directed and weighted links between two neurons i and j. Formally, a neural network is defined as follows: A NN is sorted triple (N,V,w) with two sets N, V and a function w, where N is neurons set and $V = (i,j)/i,j \in N$, these elements are connections between neuron i and j. The function $w : V \rightarrow \mathbb{R}$ defines the weight, $w((i,j))$, between i and j [13].

2.4 Processing Images

The great growth of data amount makes practically impossible for a person works all raw data and gets conclusions, trends and patterns. Data visualization techniques can help significantly to this process.

The main objective of visualization is an adequate perceptual representation of data, trends and underlying relationships between them. The visualization purpose is not only images creation but also the insight of large data amounts.

Informally, visualization is data or information transformation into images. A successful visualization can

reduce considerably the time consumed to understand data, to find relationships and to obtain information. To generate a visualization, it is necessary to map data in the Cartesian space (two or three dimensions). This space has to represent the relationships as intuitively as possible. Finding a good spatial representation of data not is a simple task, it is one of the most difficult tasks in visualization of abstract information [14].

2.5 High Performance Computing (HPC)

A Big Data Systems needs to process large data volumes and, consequently, demands greater computational capacity. For this reason, conventional computer systems are not suitable for proper processing. HPC techniques allow intensive computational operations and improve processing speed; involving different technologies such as distributed systems and parallel systems (computer clusters, cloud computing, graphic processing unit and massively parallel computers) [15, 16].

Graphical processing units (GPU) can be used to solve general purpose problems (General Purpose Graphical Processing Units, GPGPUs) [17]. When the GPU is used as a parallel computer, it necessary taken into account the processing units number, its memory structure and own programming model. The CUDA programming model, developed by NVIDIA, allows to use a GPU as a highly parallel computer, providing a high-level programming environment. It defines to GPU as a programmable co-processor of CPU with an architecture is formed by multiple streaming multiprocessors, each with several scalar processors.

In [18, 19, 20, 21], they use GPU to solve problems of massive data. They propose different techniques of data analysis in GPU. These techniques are included in areas such as machine learning, search and sorting, data mining, database, among others. This paper presents a big data problem, the anomalies detection in network traffic as soon as they happen. This work include several tasks, most of them can solve using GPU, and in consequence accelerate the solution.

3 A Big Data Problem: Analysis Network Traffic

The network traffic analysis in terms of information security is a Big Data problem. As it was said in previous sections, a problem of this type is defined by 7 Vs [22]. Anomalies or attacks detection in a Network involves to work with all circulating data in a network (volume), generated them to high speed (velocity), with heterogeneous nature (variety), belonging to normal or anomalous profiles (veracity), processed and filtered in base to significant characteristics, depending of particular system (validity), where their visual representation is possible in order to read them easily (visualization), and from them it is necessary to detect effectively and precisely attacks in a reasonable time (value).

Anomalies detection in data networks identifies unusual patterns, i.e. patterns not belong to the typical traffic in the network [23]. Detecting a possible attack requires technologies to classify traffic and associate data flows with those applications that generate them. Work data set grows at a high speed, it is much greater than processing capacity. Having the ability to handle large data volumes not only allows to know what is happening at this moment, but also to trace patterns over time.

When information is analyzed in real time, it is often easy to overlook some indicators. If information analysis is made in other context and by long time, it is possible to find other meanings. In this case, the main is to process network traffic, apply any techniques (with intelligent or not) and to obtain concrete results. All these have to be made in a short time to arrive relevant conclusions to detect real attacks and take quick decisions.

The next sections describe the first solutions developed, their advantages and drawbacks, and finally some experimental results that show their behavior.

4 First Solutions

The detecting task of possible packets with anomalous data in a computer network is very expensive. A good alternative can be combine images processing, machine learning techniques and HPC in a only one solution.

In [23, 24, 25] a detection of anomalies model to a local network called *P-SNADS* (Parallel-Supervised Network Anomalies Detection System) was presented. In figure 1, the *P-SNADS'* architecture is detailed. It has 2 stages, each of them well differentiated. These are:

- *Data Recollection:* This stage captures network traffic and organizes it in data flows. Its objective is to obtain data to work and implies the following sub-tasks:
  - Data Capture: This process catches network traffic. It uses a sniffer to do it, for example T-Shark, and it is activated at specific moments, for examples when there is more network traffic.
  - Data Selection: This step selects frames to be analyzed, according to the attacks considered, these are TCP, UDP and ICMP frames.
  - Feature Extraction: In this step, all interest fields of each frame are extracted to be analyzed. They are: source and destination IP address, source and destination port and protocol type (TCP, UDP or ICMP).

– Normalization: This step is fundamental, the tuples become an integer values vector. All address IP (IPv4) a.b.c.d are transformed according to equation (1)

$$(ax256)^3 + (bx256)^2 + (cx256)^1 + (dx256)^0 \quad (1)$$

- *Anomalies Detection*: In this stage, the task is to find possible attacks in a network. It can be solved applying different techniques. Which one? It depends on tuples number to analyze:

  – If tuple is examined at a time, the *k*-NN algorithm is applied. It compares the collected tuples with the attacks signatures.

  – If a tuples set is analyzed, different images are generated. Afterwards, similarities are searched between each created images with those representing anomalies, previously defined and stored. To compare two images, the algorithm SIFT (Scale Invariant Feature Transform) is used (It detects and describes distinctive invariant features in images. For any object in an image, interesting points (key points) on object can be extracted to provide a "feature description" of object) [26]. If these key points match, a possible attack would be detected, otherwise the current traffic is not a recognized intrusion.

The attacks signature and anomalies image repository are in constant update, they have to include new attack or new features of already known.

The above stages demand a lot of computational resources. Therefore, HPC is considered to improve both solution, particularly GPGPU (General Purpose Graphics Programming Unit). The results obtained are satisfactory, the times of the parallel solutions to each stage are significantly lower than corresponding to sequential solution, a times reduction greater than 500x is achieved.

In [24, 25], known attacks by signatures are detected by the supervised learning technique *k*-NN. Each tuple of current traffic in network, is analyzed if matches with any of known attack signatures. As result, the *K* closest to each attack are obtained, determining the occurrence or not of them.

In both cases, there is a need to have signatures or patterns constantly updated, making it impossible to detect new attacks early. This paper intends to incorporate new techniques related to machine learning without supervision, which do not require intervention of network administrator, permanently he/she has to update attacks databases. For these learning techniques, by its nature, it is also possible to apply HPC in your computational solution.

5 Experimental Results Analysis

This section presents a experimental results analysis of fist solutions of P-SNADS. To catch network data traffic, T-Shark tool is used [2]. P-SNADS works with several samples, each of them has approximately 2000 frames. These frames belong to a local area network (LAN) of Networks Laboratory of Universidad Nacional de San Luis. Three attacks are simulated in a server and they pretend to deny the HTTP service. Once obtained the normalized frames, different databases are built with normal and anomalous traffic.

The results are contrasted with sequential solution. Its times are obtained of a PC with an AMD FX (tm) -6300 Six-Core Processor x6 processor with 7.8 GB memory and an 80.5 GB disk. For HPC solution, a GPU is used. It has the following features: Tesla K20c (2496 processors), 4.6 GB of Memory and 706 MHz of clock frequency and 2600 MHz of processor memory.

The *k*-NN module receives the databases as input data and performs an evaluation for different values of *k*. Four values of *k* are considered, they are 5, 7, 10 and 12.

The Table 1 shows the necessary milliseconds to obtain *k*-NN in its two versions: sequential and parallel. In the parallel version, the time includes the transfer times of the database to the GPU, *K-NN* calculation and results transfers (the *k*-NNs) to the CPU.

Table 1: Average times (in milliseconds) of *K-NN*.

K	Time (sequential)	Time (parallel)
5	1089,90	0,0904
7	1086,60	0,0907
10	1081,20	0,0935
12	1091,60	0,0978

As can be seen, the parallel times are significantly lower to the sequential time. In addition, the parallel version shows quasi-constant behavior, independently of *k* value.

When *k*-NN are computed, the next metric are considered: Positive Predictive Value (Precision - PPV),True Positive Rate (Recall - TPR), and F-measure (F) [24]

Figure 3 shows each metric to each attack considered: Smurf (a), Land (b) and Fraggle (c).

From observation of the above graphs, Smurf attack has an approximate accuracy of 72%, while for the others two are lower: Land= 41% and Fraggle = 52%. Regarding Recall, more significant results are obtained, to Smurf is 80%, Land is 72% and Fraggle is 75%. In base of these values, it possible to infer that

[2] https://www.wireshark.org/

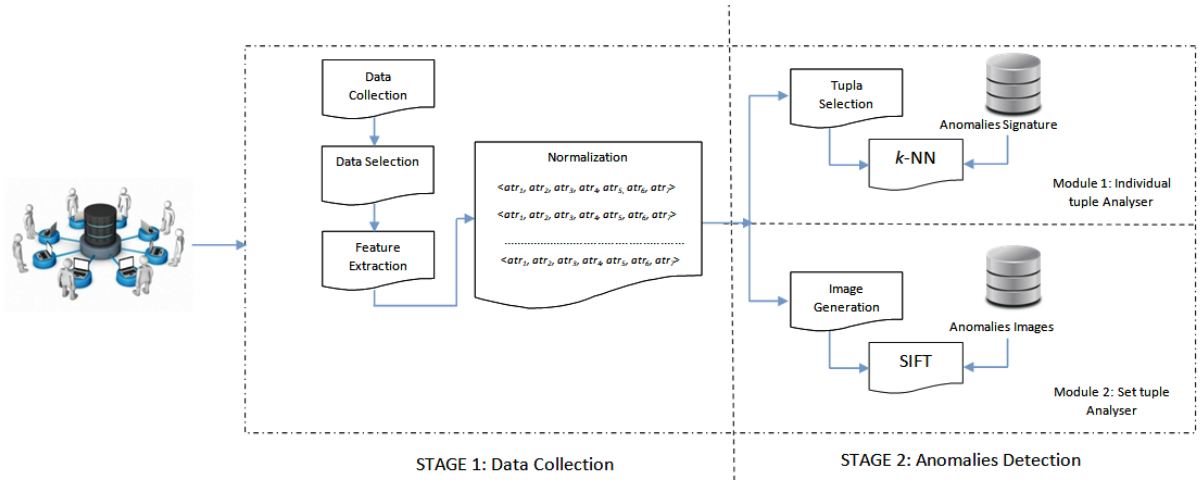


Figure 2: P-SNADS' Arquitechture

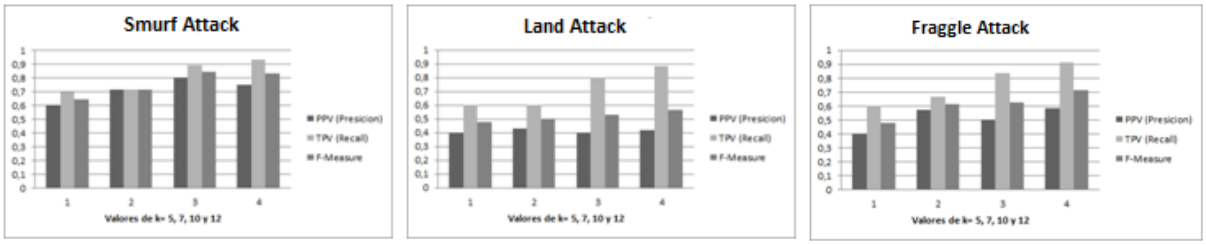


Figure 3: Results obtained in attacks: Smurf, Land and Fraggle.

the detection rate of anomalous traffic is high, particularly when k = 12. Finally, the F-measure also returns good results, particularly for k=12 no matter which attack is. The values obtained are 0.83 for Smurf; 0.56 for Land and 0.71 for Fraggle. It means that our model performs well in detecting attacks. Therefore, for metrics analyzed, P-SNADS achieves satisfactorily the objectives of this work.

For SIFT algorithm, different sizes images are considered, which sizes are between 50KB and 20MB.

The Table 2 shows how many milliseconds are needed to obtain all key points of an image using SIFT, and the Table 3 details the milliseconds quantity requires to determine the common key points between the images of traffic and the attack (this last process is known as MATCH process). In both tables, the times correspond to the sequential and parallel versions, and for all results, the time averages of several executions are displayed.

In both tables, 2 and 3, the times of parallel solutions are significantly lower than those of sequential versions, independently of image size. The times reduction is greater than 500x.

Table 2: SIFT average times in milliseconds.

Range of image size	SIFT (sequential)	SIFT (parallel)
10KB y 300KB	1780.92	14.045
300KB y 1MB	4650.075	20.615
1MB y 3MB	22084.63	64.6675
3MB y 5MB	40114.55	96.01

6 Conclusions and Future Works

In this paper, the P-SNADS model was presented, it is divided into two sequential stages with different objectives, one is the Data Collection and the other is Detection of Anomalies. This objective concentrates the greatest interest because it seeks to collaborate in the process of computer security detecting anomalous traffic in a computer network. To carry out this task, two different techniques are applied according to the nature of the attack.

When the attack has a signature, the supervised learning technique k-NN can be used, each captured traffic tupla is compared against known attacks firms. In base to results, the occurrence or not of an attack can be determined.

For attacks based in flow, tuple analysis is not a good solution. Techniques of image representation and processing are suitable. From captured traffic, different images are built in order to be able to compare with images that represent attack patterns. To establish an attack, the images comparison is made applying

Table 3: MATCH average times in milliseconds.

Range of image size	MATCH (sequential)	MATCH (parallel)
10KB y 300KB	637.09	4.635
300KB y 1MB	3325.09	8.05
1MB y 3MB	30421.66	26.885
3MB y 5MB	85513.74	71.12

SIFT algorithm.

The application of HPC techniques in several P-SADS' stages allows to improve execution times and, in consequence, early detect possible attacks to a network.

While both techniques, from the point of view of the 7 Vs, are applicable to Big Data problems, both are supervised methods, they need a constant updating of signatures or images that represent attacks.

The next step is to incorporate others Machine Learning techniques, such as Neural Networks and Deep Learning. These approaches have shown good performance in developing models and architectures for discovering patterns of malicious activities.

References

[1] Tulasi.B, R. S. Wagh, and B. S., "High performance computing and big data analytics - paradigms and challenges," *International Journal of Computer Applications*, vol. 116, Abril 2015.

[2] Y. Wang, *Statistical Techniques for Network Security: Modern Statistically-Based Intrusion Detection and Protection*. Hershey, PA: Information Science Reference - Imprint of: IGI Publishing, 2008.

[3] "S. institute, transmission control protocol: Darpa internet program protocol specification. defense advanced research projects agency, information processing techniques office," Sept. 1981.

[4] D. Gibson, "Comptia security+: Get certified get ahead: Sy0-201 study guide createspace independent pub.," 2009.

[5] H. R. J. L., "Definición de un modelo de seguridad en redes de cómputo, mediante el uso de técnicas de inteligencia artificial. tesis presentada como requisito parcial para optar al título de magíster en ingeniería – automatización industrial. universidad nacional de colombia.," 2012.

[6] I. G. B. S. B. Analytics and Optimisation, "Analytics: el uso de big data en el mundo real.," in *Escuela de Negocios Saïd en la Universidad de Oxford*.

[7] C. L. P. Chen and C. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on big data," *Inf. Sci.*, vol. 275, pp. 314–347, 2014.

[8] D. S. Terzi, R. Terzi, and S. Sagiroglu, "Big data Analytics for Network Anomaly Detection from Netflow Data," *IEEE*, 2017.

[9] A. Y. Nikravesh, S. A. Ajila, C. H. Lung, and W. Ding, "Mobile network traffic prediction using mlp, mlpwd, and svm," pp. 402–409, June 2016.

[10] T. Hind, *Análisis Estadístico de Distintas Técnicas de Inteligencia Artificial en Detecci ón de Intrusos*. PhD thesis, Universidad de Granada, 2012.

[11] N. Miranda, *Cálculo en Tiempo Real de Identificadores Robustos para Objetos Multimedia Mediante una Arquitectura Paralela GPU-CPU*. PhD thesis, Universidad Nacional de San Luis, 2014.

[12] P. M. F, *Computación de alto desempeño de GPU*. Editorial de la Universidad Nacional de La Plata (EDULP), 2011.

[13] A. M. Ghimeş and V. V. Patriciu, "Neural network models in big data analytics and cyber security," in *2017 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pp. 1–6, June 2017.

[14] S. Martig, S. Castro, M. Larrea, S. E. D. Urribarri, M. Escudero, and L. Ganuza, "Herramientas de visualización para la exploración de datos," *IX Workshop de Investigadores en Ciencias de la Computación*, 2007.

[15] G. Hager and G. Wellein, *Introduction to High Performance Computing for Scientists and Engineers*. CRC Press, Inc., 1st ed., 2010.

[16] Y. You, S. L. Song, H. Fu, A. Marquez, M. M. Dehnavi, K. J. Barker, K. W. Cameron, A. P. Randles, and G. Yang, "MIC-SVM: designing a highly efficient support vector machine for advanced modern multi-core and many-core architectures," in *2014 IEEE 28th International Parallel and Distributed Processing Symposium, Phoenix, AZ, USA, May 19-23, 2014*, pp. 809–818, 2014.

[17] NVIDIA, "Nvidia cuda compute unified device architecture, c programming guide. version 7.5," 2015.

[18] D. C. Cireşan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Mitosis detection in breast cancer histology images with deep neural networks," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2013* (K. Mori, I. Sakuma, Y. Sato, C. Barillot, and N. Navab, eds.), (Berlin, Heidelberg), pp. 411–418, Springer Berlin Heidelberg, 2013.

[19] S. Chen, J. Qin, Y. Xie, J. Zhao, and P.-A. Heng, "A fast and flexible sorting algorithm with cuda," in *Algorithms and Architectures for Parallel Processing* (A. Hua and S.-L. Chang, eds.), (Berlin, Heidelberg), pp. 281–290, Springer Berlin Heidelberg, 2009.

[20] Y. Chen, Z. Qiao, S. Davis, H. Jiang, and K.-C. Li, "Pipelined multi-gpu mapreduce for big-data processing," in *Computer and Information Science* (R. Lee, ed.), (Heidelberg), pp. 231–246, Springer International Publishing, 2013.

[21] S. Herrero-Lopez, "Accelerating svms by integrating gpus into mapreduce clusters," in *2011 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 1298–1305, Oct 2011.

[22] M. A. ud-din Khan, M. F. Uddin, and N. Gupta, "Seven v's of big data understanding big data to extract value," in *Conference of the American Society for Engineering Education*.

[23] M. Barrionuevo, M. Lopresti, N. Miranda, and F. Piccoli, "Un enfoque para la detección de anomalías en el tráfico de red usando imágenes y técnicas de computación de alto desempeño," *XXII Congreso Argentino De Ciencias de la Computación*, pp. 1166–1175, 2016.

[24] M. Barrionuevo, M. Lopresti, N. Miranda, and F. Piccoli, "An anomaly detection model in a lan using k-nn and high performance computing techniques," *Communications in Computer and Information Science*, pp. 219–230, January 2018.

[25] M. Barrionuevo, M. Lopresti, N. Miranda, and F. Piccoli, "P-sads: Un modelo de detección de anomalías en una red lan," *5to Congreso Nacional de Ingeniería Informática / Sistemas de Información Aspectos Legales y Profesionales y Seguridad Informática*, 2017.

[26] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, Nov 2004.

Adjustment of a simulator of a complex dynamic system with emphasis on the reduction of computational resources

Mariano Trigila[1], Adriana Gaudiani [2] and Emilio Luque[3]

[1] *Facultad de Ingeniería y Ciencias Agrarias, Pontificia Universidad Católica Argentina, Ciudad Autónoma de Buenos Aires, Argentina*

[2] *Instituto de Ciencias, Universidad Nacional de General Sarmiento, Buenos Aires, Argentina*

[3] *Depto. de Arquitectura de Computadores y Sistemas Operativos, Universidad Autònoma de Barcelona, 08193 Bellaterra (Barcelona) España*

Abstract

Scientists and engineers continuously build models to interpret axiomatic theories or explain the reality of the universe of interest to reduce the gap between formal theory and observation in practice. We focus our work on dealing with the uncertainty of the input data of the model to improve the quality of the simulation. To perform this type of process large volumes of data and a lot of computer processing must be handled. This article proposes a methodology for adjusting a simulator of a complex dynamic system that models the wave translation along rivers channels, with emphasis on the reduction of computation resources. We propose a simulator calibration by using a methodology based on successive adjustment steps of the model. We based our process in a parametric simulation. The input scenarios used to run the simulator at every step were obtained in an agile way, achieving a model improvement up to 50% in the reduction of the simulated data error. These results encouraged us to extend the adjustment process over a larger domain region.

1. Introduction

Scientists and engineers make use of computer simulations, as an established tool in many branches of science, to study the behavior of the system modeled. They study the model in function of the different responses of the model when different scenarios are used to run the simulation. As a preliminary act, the simulator will need an adjustment process in which the best set of input values to the simulator is sought which provides the smallest difference between the output data and the reference data set [7]. Adjustment processes are usually computationally expensive since they require running the simulator with each of the possible combinations of inputs in search of the best output. In other words, When the search space of an optimal set of parameters is very large then the computational cost of the optimal search process is very expensive. This paper presents a methodology that proposes to lower computational cost.

The proposed approach exploits a local behavior of the system: The values of the critical parameters, selected for the calibration process, differ very little in sites located spatially close to each other over the domain of the system. This assumption allows reducing the search space of the input parameter to the simulator that minimizes the error between the simulated and the real data. This artifice allows in a direct way to reduce the computational cost of the search process. Therefore, the parameters values to be optimized for each section on the river domain, are calibrated taking advantage of the optimal values which were calculated for the previous section located at an adjacent place.

Using our methodology, we could find input scenarios to run the model, which provided a substantial improvement in the quality of the prediction in relation to the results obtained when the simulation is launched with the initial scenario (currently used for simulation and forecasting). The best results obtained provided a gain of up to 50%. To determine this value, we detected one input parameters set used to launch a simulation which is the one that best fits for a predetermined sampling site located on the riverbed. We calculated an index to quantify the difference between the simulated and the observed series of data. The search process ends when it finds the best parameters set, by which we mean the scenario that gets the lowest index. Therefore, the scenario obtained is the best simulation scenario in a reduced search space. We take advantage of the research and the results of previous works [2, 3].

2. The simulator and the simulation domain

The simulator implements a one-dimensional hydrodynamic model of the Paraná River for hydrological forecast [4, 5]. This computer model calculates the translation of the waves through a channel calculated by the Saint Venant equations. It

was developed in the Laboratory of computational hydraulics of the National Institute of water (INA).

The hydrodynamic model simulates a physical system whose domain is set by its parameters values. In summary, the simulator could be described as an "input - process - output" system [6], where: The input is a complete simulation scenario, including the set of parameters and the input data needed by the model to simulate the behavior of the river, which is the simulated physical system. The process is defined by the algorithms of the computational model which relate the system variables and its evolution. This model is based on numerical methods that solve equations in partial derivatives. The simulator output is the set of simulated data returned by the model, when it was executed with a given input scenario. Some of the parameters and inputs that define the system and the simulation domain are the following: Input - height of levees, channel Manning, plain Manning, flow, border and start conditions, among others. The input parameters and the input variables required for system initialization are stored in text files. Process - Arithmetic calculations, algorithms, procedures and functions to resolve the river wave's displacement, are implemented in a Fortran program. Output - the river height, flow, among others, calculated at the monitoring stations. The output data are stored in text files.

3. Domain modeling features

Simulator represents a hydrodynamic model consisting of two sections or filaments. Each filament represents the path of a river. See data in Table 1 and a graphical representation in Fig. 1. To simulate the transport of water in a filament channel, its route is subdivided into sections. Each section (Sc) that divides the domain results from a discretization process and represents a specific position over river path, as it is shown in Fig. 1.

Table 1 Calculation Network: River model [3].

#	Path	*Long. (km)*	*Sections*
5	Paraná	1083	76
	Paraguay	376	77

The simulator requires setting a set of input parameters values at every subsection in each section. Each set determines a simulation scenario. At the same time, each section has a subdivision called subsection (Su). Each subsection is a cross section to the channel, and it describes the geometry of the river in a section. The set of input parameters of a section is composed of the subsections parameters that it has. For every subsection in each section a set of parameters is specified, of which we consider for this work the roughness coefficient of Manning (m), which varies according to the resistance offered by the main channel and the floodplain, being necessary to distinguish them at a value of Manning of plain (mp), and Manning of channel (mc).

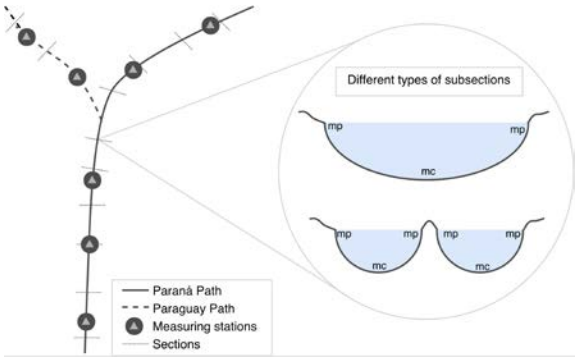


Fig. 1 Discretization of the simulated domain river system.

Depending on the channel geometry in each section, a greater or lesser amount is needed to mp and mc. The different sections can be seen in Fig. 1.

3.1. Observed data measured at monitoring stations

A monitoring or measuring station ($\boldsymbol{St}$) is the "physical and real" place where the river heights are surveyed and recorded. A measuring station is in a city on the banks of the river channel. The data collected and recorded from the height of the river are known as observed data ($\boldsymbol{OD}$) and are measured daily. The period from 1994 to 2011 is available for all monitoring stations and these data were used to implement the experiences carried out in this work.

3.2. Observed data vs. Simulated data

At the beginning of the problem analysis, we concentrate on finding the difference, or simulation error, between the observed data series and the simulated data series. To show these differences we used data visualization techniques, among which is the "Stream Graph" Fig. 2 [9].

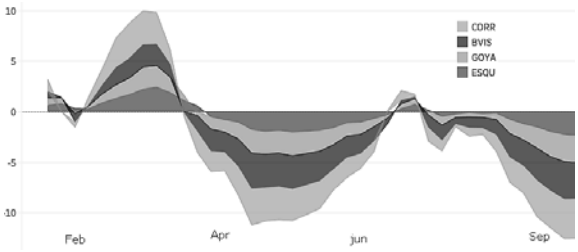


Fig. 2 Difference Observed - Simulated 1997, four stations.

With it we were able to observe the pronounced differences over time, comparing four stations. We

could also observe the relationship of similarity between series of stations. The graphed stations are spatially contiguous to one another.

4. Methodology

We propose a calibration process of successive tuning steps to obtain an adjusted input parameters values from a preselected set of successive sections. The process varies the input parameters values in a preset way as we explained in section 4.2. An entire simulation scenario must be used to feed the simulator for each of the possible combinations of parameters values. Each combination determines a simulation scenario and we detail the input scenario structure later. The quality of the simulated data (SD) is measured through calculating a divergence index (DI), as we explain in section 4.4. We propose a search methodology for finding the best set of parameters, to optimize the simulation for a reduced search space ω such that ω ⊂ Ω, therefore, minimizing the use of computing resources to achieve the objective, min (DI); where Ω is the whole search space with all possible combinations of the selected adjustment parameters and ω is the resulting reduced space [8]. We show in Fig. 3 the implemented process to search the adjusted parameters set $\hat{X}$, for the station k, which determines the best simulation scenario $\widehat{S_k}$. We start the method by choosing a monitoring station St_k located in an arbitrary place k on the riverbed and selecting three contiguous sections, which are adjacent to that station. After obtaining the best scenario for a station in k, the tuning method is successively extended to its adjacent stations in k+1, repeating the search and successive adjustment process for the n stations, as we show in Fig. 4.

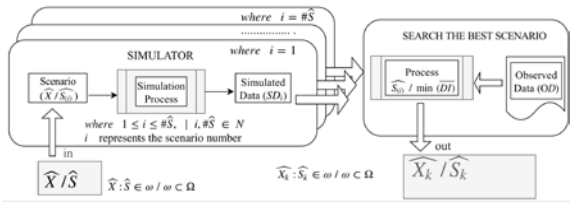


Fig. 3 Search process of the best scenario

4.1. Selecting stations and sections

We chose the first monitoring station St_k which will be the first station located upstream on the river. This is convenient that the chosen place has a simple geometry and that there are measured observed data. The following station to be adjusted, and the next ones, will be chosen by the adjacency to a previously adjusted station. Once selected the first station St_k was selected, we choose a group of sections (Sc_i) located in an adjacent way to the first one station St_k, chosen in the previous step. Three sections surrounding the station k were selected, to carry on for the experiences. They are, the m section, that matches the location of St_k, a second section Sc_{m+1} located adjacent and upstream to St_k and a third section Sc_{m+2} located adjacent and downstream to the section St_k. For this work, we selected simple geometry sections, with three or five subsections, Su.

4.2. Structure of the input scenario

The friction parameter, Manning coefficient, is set for each subsection (Su_m), as we explain in next section. In this case, three subsections determine the section Su_m chosen. Therefore, a section is defined by an j-tuple of Su_m. For this case, the subsections that describe the section Sc_m is defined by 3-tuple:

$$Sc_m = (Su_{J(1)}, Su_{J(2)}, Su_{J(3)}) \quad \text{(Eq. 1)}$$

where each $Su_{J(x)}$ is defined by a Manning coefficient. $Su_{J(1)}$ and $Su_{J(3)}$ are defined by the same Manning of plain mp_m, and $Su_{J(2)}$ by a Manning of channel mc_m. Thus, Sc_m can be represented by 3-tuple based on Manning coefficients.

$$Sc_m = (mp_m, mc_m, mp_m) \quad \text{(Eq. 2)}$$

We remark that equation (Eq. 2) has two independent variables, mp_m and mc_m. For k station, three contiguous and adjacent sections were chosen, as we explained previously. The scenario $\widehat{S_k}$ for station k will be defined by:

$$\widehat{S_k} = \begin{bmatrix} Sc_m \\ Sc_{m+1} \\ Sc_{m+2} \end{bmatrix} = \begin{bmatrix} mp_m & mc_m & mp_m \\ mp_{m+1} & mc_{m+1} & mp_{m+1} \\ mp_{m+2} & mc_{m+2} & mp_{m+2} \end{bmatrix} \text{(Eq. 3)}$$

Being a physical system, and because the sections are close together, it is assumed that the three sections have the same values of mp y mc for St_k. Summarizing, equation (Eq. 3) results in:

$$\widehat{S_k} = \begin{bmatrix} Sc_m \\ Sc_{m+1} \\ Sc_{m+2} \end{bmatrix} = \begin{bmatrix} mp_k & mc_k & mp_k \\ mp_k & mc_k & mp_k \\ mp_k & mc_k & mp_k \end{bmatrix} \text{(Eq. 5)}$$

We remark in equation (Eq. 4) that, mp_k and mc_k are independent variables. Therefore, the input scenario used to start the tuning process $\hat{X}$ is determined by the scenarios $\widehat{S_k}$ corresponding to the sections Sc_m, and for the intermediate scenarios $\widehat{S_k}^+$ corresponding to the intermediate sections ${Sc_m}^+$ located between the stations k y $k+1$. Equation (Eq. 5) represents $\hat{X}$ structure for n stations:

$$\widehat{X} = \left\{\widehat{S_k}, \widehat{S_k}^{+}, \widehat{S_{k+1}}, \widehat{S_{k+1}}^{+}, \ldots, \widehat{S_n}\right\},$$

$$with\ k = 1 \text{ (Eq. 6)}$$

4.3. Manning variation range.

We had to set the values of the variation range of plain and channel Manning coefficient, imp and imc, and their corresponding increment value, smp and smc. Both determine the discretization process when determining the parameters values.

$$imp = \left[mp_{min,}\, mp_{max}\right] = [0.1,\ 0.71];$$

$$smp = 0.01 \text{ (Eq. 7)}$$

$$imc = \left[mc_{min,}\, mc_{max}\right] = [0.017, 0.078];$$

$$smc = 0.001 \text{ (Eq. 8)}$$

$$\#\hat{S} = 61 \mid \frac{mp_{max} - mp_{min,}}{smp} = \#\widehat{S} \ \wedge \ \frac{mc_{max} - mc_{min,}}{smc} = \#\widehat{S} \text{ (Eq. 9)}$$

The value 61, for the number of scenarios ($\#\widehat{S}$), was obtained empirically after making previous experiences and finding a minimum value of scenarios which allow us to get improved output values when running the simulation. Of course, we can increment $\#\widehat{S}$ if more precision is required but this requirement will result in the need for many more computational resources. Equation (Eq. 9) determines the values of each scenario $\widehat{S_{k(\iota)}}$ depending on the selected step:

$$\widehat{S_{k(\iota)}} = \begin{bmatrix} mp_i & mc_i & mp_i \\ mp_i & mc_i & mp_i \\ mp_i & mc_i & mp_i \end{bmatrix} =$$

$$\begin{bmatrix} (smp \cdot i) + mp_{ini} & (smc \cdot i) + mc_{ini} & (smp \cdot i) + mp_{ini} \\ (smp \cdot i) + mp_{ini} & (smc \cdot i) + mc_{ini} & (smp \cdot i) + mp_{ini} \\ (smp \cdot i) + mp_{ini} & (smc \cdot i) + mc_{ini} & (smp \cdot i) + mp_{ini} \end{bmatrix}$$

(Eq. 10)

Where i is the number of scenario and the range $\left[i, \#\widehat{S}\right] \subset \mathbb{N}, where\ 1 \leq i \leq \#\widehat{S}$. mp_{ini} and mc_{ini} are the initial values used to start the search process and running the simulator for each scenario to find the best one, as we describe in next section

4.4. Search of the best scenario

When the simulator is fed with each of the possible experimentation scenarios, the output produces a numerical series of simulated data (SD), which are used to generate hydrographs of the riverbed heights. We select those series corresponding to the chosen station to implement our fitness functions by comparing the SD series with the OD series. A divergence index DI is determined, and is implemented using the root mean square error estimator (RMSE):

$$\boldsymbol{DI_k^y} = RSME_k^y = \sqrt[2]{\frac{\sum_{i=1}^{i=N} (H_k^{OD,y} - H_k^{SD,y})_i^2}{\mathbf{N}}} \text{ (Eq. 11)}$$

The index DI_k^y is calculated based on the RMSE error of the series of river heights simulated $H_k^{SD,y}$ with respect to of the series of river heights observed $H_k^{OD,y}$, for a station k, and for a year y, which is the simulation time, and the number of stations, N. Every time a simulation ends, we evaluate $DI_{k\ i}^y$ for each scenario, which are indicated by the i sub index. The best fit scenario for the k station is denominated $\widehat{S_{k(\iota)}}$ which generates a set of output H_k^{SD} such that DI_y^k is the minimum $(\min(\overline{DI_k^y}))$ of all the simulations. So, $\widehat{S_k}$ is denoted by the "best fit" scenario for station k where $\propto$ represents the sub index that best fits.

4.5. Successive tuning process

After obtaining the scenario of best fit for a station, the adjustment can be extended to a new station $k + 1$ taking advantage of the locality simulation behavior and the parameters set values of a previously adjusted station k, which is the neighboring station to the $k + 1$ station. For it, the scenario $\widehat{S_k}^{+}$ is initialized with the values of the best fit scenario $\widehat{S_k}$. This is so because by locality behavior, those sections that are close one to another have similar adjustment scenarios or at least differ very little abrupt jumps (or changes) in parameter values in distinct positions of the selected stretch of river. Therefore, we are already able to run the simulator and look for the best scenario $\widehat{S_{k+1(\propto)}}$ for the station $k + 1$. We can see details in Fig. 4.

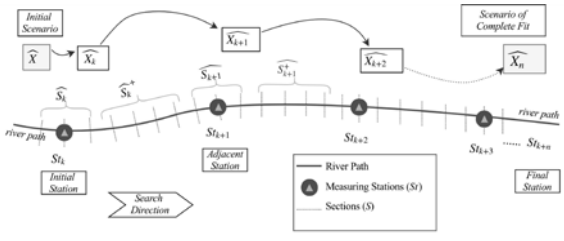


Fig. 4 Methodological Successive tuning process.

In the successive input scenarios, we leave fixed the adjusted parameters values found in the previous calibrations, and thus the previous adjustment scenarios of each section are used to find de actual adjusted parameters values, for k station the new k parameter vector is:

$$\widehat{X_k} = \left\{\widehat{S_k}, \widehat{S_k}^+, \widehat{S_{k+1}}, \widehat{S_{k+1}}^+, \ldots., \widehat{S_n}\right\}$$

The $k+1$ input scenario, or k+1 parameters vector, is:

$$\widehat{X_{k+1}} = \left\{\widehat{S_k}, \widehat{S_k}^+, \widehat{S_{k+1}}, \widehat{S_{k+1}}^+, \ldots., \widehat{S_n}\right\},$$
$$where\ \widehat{S_k}^+ = \widehat{S_k}$$

The $k+2$ input scenario, or k+2 parameters vector is:

$$\widehat{X_{k+2}} = \left\{\widehat{S_k}, \widehat{S_k}^+, \widehat{S_{k+1}}, \widehat{S_{k+1}}^+, \ldots, \widehat{S_n}\right\},$$
$$where\ \widehat{S_k}^+ = \widehat{S_k}, \quad \widehat{S_{k+1}}^+ = \widehat{S_{k+1}}$$

For n input scenario to the Simulator (scenario that adjusts the entire domain):

$$\widehat{X_n} = \left\{\widehat{S_k}, \widehat{S_k}^+, \widehat{S_{k+1}}, \widehat{S_{k+1}}^+, \ldots, \widehat{S_n}\right\},$$
$$where\ \widehat{S_k}^+ = \widehat{S_k}, \ldots, \widehat{S_{n-1}}^+ = \widehat{S_{n-1}} \quad \text{(Eq. 12)}$$

4.6. Experimental Results

After making the experiences, feeding the simulator with the proposed scenarios, and analyzing and comparing the series of outputs delivered by the simulator against the observed series, positive results were obtained in terms of meeting scenarios of better performance than the initial proposed by experts in the domain of the problem.

In search of the best scenario performed on the k station "Esquina" (ESQU), we found scenarios that improved the output of the simulator up to 57% in relation to the initial scenario proposed by the experts in the domain of the problem, determined by ratio of $DI_k^y(Fit)$ to $DI_k^y(Initial)$. We show in Table 2 the synthesis process with the top three scenarios found for processed k station. As also, it shows the second station $k+1$ adjusted. The best scenario is searched at "La Paz" station (LAPA) which is adjacent to ESQU station. As it can be observed in Table 2, a synthesis process with the two best scenarios was found for $k+1$ station.

Table 2 Fit made in k station and k+1 station, several years.

S_i	*Year*	*Station*	*Station ID*	*Improvement* $DI_k^y(F)/DI_k^y(I)$
46	*2008*	*k*	*ESQU*	*57 %*
54	1999	*k*	ESQU	39 %
38	2002	*k*	ESQU	22 %
38	1999	*k+1*	LAPA	45 %
30	2008	*k+1*	LAPA	24 %

Fig. 5 and Fig. 6 show a comparative graph with the observed data series (real measured values), the initial simulated data series (original series loaded in the simulator) and the series of simulated data adjusted for the best fit scenarios in each (k *and* $k+1$) station. We can see that our method achieves the best results since month 4 to 12, when the simulation errors decrease.

The key to the method for the reduction of computational resources lies in:

1. To assign to $\widehat{S_k}^+$ the same value as $\widehat{S_k}$ based on the local behavior of the system.
2. To run a parametric simulation for every parameter value combination in the reduced Search space reduction $\omega \subset \Omega$.

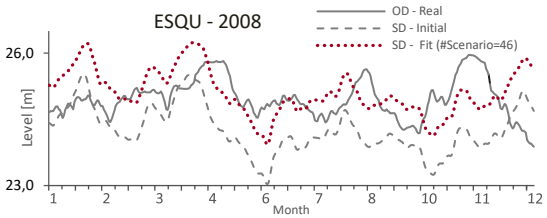


Fig. 5 Comparative OD, SD, Fit (ESQU)

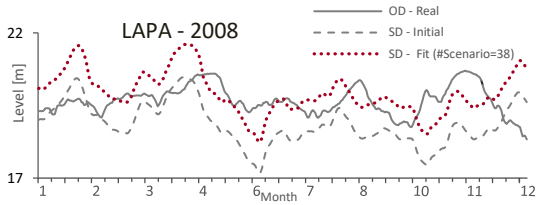


Fig. 6 Comparative OD, SD, Fit (LAPA)

5. Results of experience

In search of the best fit, in one of the stations "Esquina" (ESQU), we found a scenario (*SD – Fit #Scenario=46*) that improves the results up 57% in relation to the initial scenario used by the of INA experts, (*SD - Initial*), as you can see in Fig. 3, where it is also related to OD (*OD - Real*).

The gain value (quality) is obtained by dividing the $DI_{inicial}$ respect of DI_{Fit}.

We used the same methodology to tune forecasting for LAPA, which is a contiguous station to ESQU, getting improvements, as you can see in Table 2. Other stations that were adjusted (and not found in table 2) showed substantial improvements in error reduction.

These promising results indicate the great potential of our successive tuning steps method and encourage us to continue our research in this direction.

6. Conclusions

The main result of this work was to reduce the simulation error of the computational model using

the local properties of the river channel, in order to reduce the search space of its optimal set of parameters. The method provides promising results by finding higher quality scenarios with improvements close to the 50 %. The method is simple and manages to reduce computational resources by lowering the volume of data processed in each stage by the following fundamental reasons:

1) Each successive adjustment step results in adjusted sections. These adjusted sections will be useful for the configuration of the previous sections to a station to be adjusted (contiguous), in a new adjustment step.
2) We assume that the value of the adjustment scenario of intermediate section ($\widehat{S_k}^+$) is equal to the value of the previous immediate contiguous adjustment, by the principle of location of the system.
3) Significate reduction of search space, used to find the adjusted set of input parameters to the simulator.

We observe that the search process of the best scenario, Fig. 3, is about an embarrassingly parallel problem. Consequently, we are currently working on the implementation of the adjustment method in successive steps on HPC cloud computing platform [10, 11].

7. Acknowledgments

The MICINN/MINECO Spain under contracts TIN2014-53172-P and TIN2017-84875-P has supported this research. We are very grateful for the data provided by INA and we appreciate the guidance received from researchers at INA Hydraulic Laboratory.

8. References

[1] E. Bladé, M. Gómez-Valentín, J. Dolz, J. L. Aragón-Hernández, G. Corestein, y M. Sánchez-Juny, “Integration of 1D and 2D finite volume schemes for computations of water flow in natural channels”, Advances in Water Resources, vol. 42, pp. 17–29, 2012.

[2] E. Cabrera, E. Luque, M. Taboada, F. Epelde, y M. L. Iglesias, “Optimization of emergency departments by agent-based modeling and simulation”, in Information Reuse and Integration (IRI), 2012 IEEE 13th International Conference on, 2012, pp. 423–430.

[3] A. Gaudiani, E. Luque, P. García, M. Re, M. Naiouf, y A. De Giusti, “How a Computational Method Can Help to Improve the Quality of River Flood Prediction by Simulation”, in Advances and New Trends in Environmental and Energy Informatics, Springer, 2016, pp. 337–351.

[4] A. N. Menéndez, “Three decades of development and application of numerical simulation tools at INA Hydraulics Lab”, en First South-American Congress on Computational Mechanics, Santa Fe-Paraná, Argentina, 2002.

[5] T. Krauße and J. Cullmann, “Identification of hydrological model parameters for flood forecasting using data depth measures”, Hydrology & Earth System Sciences Discussions, vol. 8, n.o 2, 2011.

[6] D. P. Solomatine and A. Ostfeld, “Data-driven modelling: some past experiences and new approaches,” Journal of hydroinformatics, vol. 10, no. 1, pp. 3–22, 2008.

[7] R. G. Sargent, “Verification and validation of simulation models,” in Simulation Conference, 2007 Winter, 2007, pp. 124–137.

[8] W. Long-Fei and S. H. I. Le-Yuan, “Simulation optimization: a review on theory and applications,” Acta Automatica Sinica, vol. 39, no. 11, pp. 1957–1968, 2013.

[9] J. Heer, M. Bostock, and V. Ogievetsky, “A tour through the visualization zoo,” Queue, vol. 8, no. 5, p. 20, 2010.

[10] D. Sikeridis, I. Papapanagiotou, B. P. Rimal, and M. Devetsikiotis, “A Comparative Taxonomy and Survey of Public Cloud Infrastructure Vendors,” arXiv preprint arXiv:1710.01476, 2017.

[11] G. Fox, J. Qiu, S. Jha, S. Ekanayake, and S. Kamburugamuve, “Big data, simulations and hpc convergence,” in Big Data Benchmarking, Springer, 2015, pp. 3–17.

Towards Elastic Virtual Machine Placement in Overbooked OpenStack Clouds under Uncertainty

Fabio López-Pires[1], Benjamín Barán[2], Carolina Pereira[2], Marcelo Velázquez[2], and Osvaldo González[2]

[1]*Itaipu Technological Park, Hernandarias, Paraguay*
fabio.lopez@pti.org.py
[2]*National University of the East, Ciudad del Este, Paraguay*
{bbaran,cpereira,mvelazquez,ogonzalez}@fpune.edu.py

Abstract

Cloud computing datacenters currently provide millions of virtual machines in highly dynamic Infrastructure as a Service (IaaS) markets. As a first step on implementing algorithms previously proposed by the authors for Virtual Machine Placement (VMP) in a real-world IaaS middleware, this work presents an experimental comparison of these algorithms against current algorithms considered for solving VMP problems in OpenStack. Several experiments considering scenario-based simulations for uncertainty modelling demonstrate that the proposed algorithms present promising results for its implementation towards real-world operations. Next research steps are also summarized.

Keywords: Virtual Machine Placement, OpenStack, Multi-Objective Optimization, Cloud Datacenters.

1 Introduction

This work focuses on a well-known problem: the process of selecting which requested virtual machines (VMs) should be hosted at each available physical machine (PM) of a cloud computing infrastructure, denoted in the specialized literature as Virtual Machine Placement (VMP). A previously proposed complex Infrastructure as a Service (IaaS) environment for VMP problems is considered, taking into account service elasticity and overbooking of physical resources [1].

In this context, this work also considers a previously proposed two-phase optimization scheme, decomposing the VMP problem into two different sub-problems, combining advantages of online (incremental VMP or iVMP) and offline (VMP reconfiguration or VMPr) VMP formulations. This is mainly because online decisions made along the operation of a dynamic cloud computing infrastructure negatively affects the quality of obtained solutions in VMP problems when comparing to offline decisions [2]. Unfortunately, offline VMP formulations are not appropriate for highly dynamic real-world IaaS environments, where cloud services are requested according to current demand.

When studying a two-phase optimization scheme for VMP problems, additional considerations should be analysed, e.g. methods to decide when or under what circumstances to trigger placement reconfigurations with migration of VMs between PMs (*VMPr Triggering*) and what to do with cloud services requested during placement recalculation (*VMPr Recovering*).

Due to the randomness of customer requests, VMP problems should be formulated under uncertainty [3]. This work considers a scenario-based uncertainty approach for modeling relevant uncertain parameters.

Taking into account experimental results already obtained in simulations against state-of-the-art alternative approaches for VMP problems considering 400 experimental scenarios, the implementation of the already proposed algorithms in a real-world IaaS middleware is a natural continuation of the work presented in [1]. As a previous step of the mentioned implementation, this work considers a previously developed Dynamic VMP Framework[1] for extending simulations including current VMP algorithms considered in OpenStack. The official OpenStack Filter Scheduler algorithm was slightly adapted to fit into the considered formulation, as described in the following sections.

The remainder of this paper is structured in the following way: Section 2 presents the considered uncertain VMP problem formulation, while Section 3 presents details on the design and implementation of evaluated alternatives to solve the formulation of the VMP problem. Section 4 summarize experimental results. Conclusions and future work are left to Section 5.

2 Considered VMP Formulation

This section summarizes the considered VMP formulation under uncertainty previously proposed by some of the authors in [1]. This VMP formulation is based on a two-phase scheme for the optimization of the following objective functions: (i) power consumption, (ii) economical revenue, (iii) resource utilization and (iv) placement reconfiguration time.

[1]http://*github.com/DynamicVMP*

According to the taxonomy presented in [4], this work focuses on a provider-oriented VMP for federated-cloud deployments, considering a combination of two types of formulations: (i) online (i.e. iVMP) and (ii) offline (i.e. VMPr). Interested readers may refer to [1] for more details on the motivation of using a two-phase optimization scheme as well as more details on the VMP formulation itself that are not included in this work due to space limitations.

The following sub-sections summarize the most relevant details on the considered uncertain VMP formulation previously proposed in [1].

2.1 Complex IaaS Environment

The considered formulation of the VMP problem models a complex IaaS environment, composed by available PMs and VMs requested at each discrete time t, considering the following information as input data for the proposed VMP problem:

- a set of n available PMs and specifications (1);
- a set of $m(t)$ VMs requested, at each discrete time t, and specifications (2);
- information about the utilization of resources of each active VM at each discrete time t (3);
- current placement at each discrete time t (i.e. $x(t)$) (4).

The iVMP and VMPr sub-problems consider different sub-sets of the above mentioned input data, as presented later in Sections 2.2.1 and 2.3.1.

The set of PMs owned by the IaaS provider is represented as a matrix $H \in \mathbb{R}^{n\times(r+2)}$, as presented in (1). Each PM H_i is represented by r different physical resources. This work considers $r = 3$ physical resources (Pr_1 to Pr_3): CPU [EC2 Compute Unit (ECU)], RAM [GB] and network capacity [Mbps]. The maximum power consumption [W] is also considered. Finally, considering that an IaaS provider could own more than one cloud datacenter, PMs notation also includes a datacenter identifier c_i, i.e.

$$H = \begin{bmatrix} Pr_{1,1} & \dots & Pr_{r,1} & pmax_1 & c_1 \\ \dots & \dots & \dots & \dots & \dots \\ Pr_{1,n} & \dots & Pr_{r,n} & pmax_n & c_n \end{bmatrix} \quad (1)$$

where:

$Pr_{k,i}$:	Physical resource k on H_i, where $1 \leq k \leq r$;
$pmax_i$:	Maximum power consumption of H_i in [W];
c_i:	Datacenter identifier of H_i, where $1 \leq c_i \leq c_{max}$;
n:	Total number of PMs.

In this context, the IaaS provider dynamically receives requests of cloud services for placement (i.e. a set of inter-related VMs) at each discrete time t. A cloud service S_b is composed by a set of VMs.

The set of VMs requested by customers at each discrete time t is represented as a matrix $V(t) \in \mathbb{R}^{m(t)\times(r+2)}$, as presented in (2). In this work, each VM V_j requires $r = 3$ different virtual resources ($Vr_{1,j}(t)$-$Vr_{3,j}(t)$): CPU [ECU], RAM memory [GB] and network capacity [Mbps]. Additionally, a cloud service identifier b_j is considered, as well as an economical revenue R_j [$] associated to each VM V_j.

$$V(t) = \begin{bmatrix} Vr_{1,1}(t) & \dots & Vr_{r,1}(t) & b_1 & R_1(t) \\ \dots & \dots & \dots & \dots & \dots \\ Vr_{1,m(t)}(t) & \dots & Vr_{r,m(t)}(t) & b_m(t) & R_{m(t)}(t) \end{bmatrix} \quad (2)$$

where:

$Vr_{k,j}(t)$:	Virtual resource k on V_j, where $1 \leq k \leq r$;
b_j:	Service identifier of V_j;
$R_j(t)$:	Economical revenue for allocating V_j in [$] at instant t;
$m(t)$:	Number of VMs at each discrete time t, where $1 \leq m(t) \leq m_{max}$;
m_{max}:	Maximum number of VMs.

To model a dynamic VMP environment taking into account both vertical and horizontal elasticity of cloud services, as previously presented in [1], the set of requested VMs $V(t)$ may include the following types of requests for cloud service placement at each time t:

- **cloud services creation:** where new a cloud service S_b, composed by one or more VMs V_j, is created. Consequently, the number of VMs at each discrete time t (i.e. $m(t)$) is a function of time;
- **scale-up / scale-down of VMs resources:** where one or more VMs V_j of a cloud service S_b increases (scale-up) or decreases (scale-down) its capacities of virtual resources with respect to current demand (vertical elasticity). In order to model these considerations, virtual resource capacities of a VM V_j (i.e. $Vr_{1,j}(t)$-$Vr_{3,j}(t)$) are a function of time, as well as the associated economical revenue ($R_j(t)$);
- **cloud services scale-out / scale-in:** where a cloud service S_b increases (scale-out) or decreases (scale-in) the number of associated VMs according to current demand (horizontal elasticity). Consequently, the number of VMs V_j in a cloud service S_b at each discrete time t, denoted as $mS_b(t)$, is a function of time;
- **cloud services destruction:** where virtual resources of cloud services S_b, composed by one or more VMs V_j, are released.

Resource utilization of each VM V_j at each discrete time t is represented as a matrix $U(t) \in \mathbb{R}^{m(t)\times r}$, as presented in (3):

$$U(t) = \begin{bmatrix} Ur_{1,1}(t) & \dots & Ur_{r,1}(t) \\ \dots & \dots & \dots \\ Ur_{1,m(t)}(t) & \dots & Ur_{r,m(t)}(t) \end{bmatrix} \quad (3)$$

where:

$Ur_{k,j}(t)$: Utilization ratio of $Vr_k(t)$ in V_j at each discrete time t.

The current placement of VMs into PMs ($x(t)$) represents VMs requested in the previous discrete time $t-1$ and assigned to PMs; consequently, the dimension of $x(t)$ is based on the number of VMs $m(t-1)$. The placement at each discrete time t is represented as a matrix $x(t) \in \{0,1\}^{m(t-1)\times n}$, as defined in (4):

$$x(t) = \begin{bmatrix} x_{1,1}(t) & x_{1,2}(t) & \dots & x_{1,n}(t) \\ \dots & \dots & \dots & \dots \\ x_{m(t-1),1}(t) & x_{m(t-1),2}(t) & \dots & x_{m(t-1),n}(t) \end{bmatrix} \quad (4)$$

where:

$x_{j,i}(t) \in \{0,1\}$: indicates if V_j is allocated ($x_{j,i}(t) = 1$) or not ($x_{j,i}(t) = 0$) for execution in a PM H_i at time t (i.e. $x_{j,i}(t) : V_j \rightarrow H_i$).

2.2 Incremental VMP (iVMP)

In online algorithms for solving the considered VMP problem, placement decisions are performed at each discrete time t. The formulation of the considered iVMP (online) problem is based on [2] and could be formally enunciated as:

Given a complex IaaS environment composed by a set of PMs (H), a set of active VMs already requested before time t ($V(t)$), and the current placement of VMs into PMs (i.e. $x(t)$), it is sought an incremental placement of $V(t)$ into H for the discrete time $t+1$ ($x(t+1)$) without migrations, satisfying the problem constraints and optimizing the considered objective functions.

2.2.1 Input Data for iVMP

As presented in [1], the considered formulation of the iVMP problem receives the following information as input data:

- a set of n available PMs and specifications (1);
- a dynamic set of $m(t)$ requested VMs (already allocated VMs plus new requests) and specifications (2);
- information about the utilization of resources of each active VM at each discrete time t (3);
- current placement at each discrete time t (i.e. $x(t)$) (4).

2.2.2 Output Data for iVMP

The result of the iVMP phase at each discrete time t is an incremental placement $\Delta x(t)$ for the next time instant in such a way that $x(t+1) = x(t) + \Delta x(t)$. Clearly, the placement at $t+1$ is represented as a matrix $x(t+1) \in \{0,1\}^{m(t)\times n}$, as defined in (5):

$$x(t+1) = \begin{bmatrix} x_{1,1}(t+1) & x_{1,2}(t+1) & \dots & x_{1,n}(t+1) \\ \dots & \dots & \dots & \dots \\ x_{m(t),1}(t+1) & x_{m(t),2}(t+1) & \dots & x_{m(t),n}(t+1) \end{bmatrix} \quad (5)$$

Formally, the placement for the next time instant $x(t+1)$ is a function of the current placement $x(t)$ and the active VMs at discrete time t, i.e.:

$$x(t+1) = f[x(t), V(t)] \quad (6)$$

2.3 VMP Reconfiguration (VMPr)

As it was previously mentioned in [1] an offline algorithm solves a VMP problem considering a static environment where VM requests do not change over time and considers migration of VMs between PMs. The formulation of the proposed VMPr (offline) problem is based on [5, 6] and could be enunciated as:

Given a current placement of VMs into PMs ($x(t)$), it is sought a placement reconfiguration through migration of VMs between PMs for the discrete time t (i.e. $x'(t)$), satisfying the constraints and optimizing the considered objective functions.

2.3.1 Input Data for VMPr

The proposed formulation of the VMPr problem receives the following information as input data:

- a set of n available PMs and specifications (1);
- information about the utilization of resources of each active VM at discrete time t (3);
- current placement at discrete time t (i.e. $x(t)$) (4).

2.3.2 Output Data for VMPr

The result of the VMPr problem is a placement reconfiguration through migration of VMs between PMs for the discrete time t (i.e. $x'(t)$), represented by:

- a placement reconfiguration of $x(t)$, i.e. $x'(t)$ (4);

Summarizing the considered constraints, a VM V_j must be allocated to run on a single PM H_i or alternatively located in another federated IaaS provider. It should be mentioned that from an IaaS provider perspective, elastic cloud services usually are considered more important than non-elastic ones. Consequently, resources of elastic cloud services most of the time are allocated with higher priority over non-elastic ones,

what usually is reflected in the contracts between an IaaS provider and each customer. Additionally, a PM H_i must have sufficient available resources to meet the dynamic requirements of all VMs V_j that are allocated to run on H_i. It is important to remember that resources of VMs are dynamically used, giving space to re-utilization of idle resources that were already reserved. Re-utilization of idle resources could represent higher risk of unsatisfied demand in case utilization of resources increases in a short period of time. Therefore, providers need to reserve a percentage of idle resources as a protection (defined by a protection factor λ_k) in case overbooking is used.

2.4 Objective Functions

More than 60 different objective functions for VMP problems were already identified in [4, 7]. Considering the large number of existing objective functions, identified objective functions with similar characteristics and goals could be classified into 5 objective function groups [4]: (G1) energy consumption, (G2) network traffic, (G3) economical costs, (G4) resource utilization and (G5) performance.

As previously considered in [1], the optimization of four objective functions is taken into account. It is important to consider that by no means, the authors claim that the considered objective functions represent the best way to model VMP problems. This formulation only illustrates a reasonable formulation of a VMP problem in order to be able to study the main contributions of this work, considering the presented experimental evaluation of VMP algorithms.

In general, objective functions can be minimized while maximizing other objectives functions. In this work each considered objective function is formulated in a single optimization context (i.e. minimization).

2.4.1 Power Consumption Minimization

The power consumption minimization can be represented by the sum of the power consumption of each PM H_i that composes the complex IaaS environment (see Section 2.1), as defined in (7).

$$f_1(x,t) = \sum_{i=1}^{n} ((pmax_i - pmin_i) \times Ur_{1,i}(t) + pmin_i) \times Y_i(t) \quad (7)$$

where:

- x: Evaluated solution of the problem;
- $f_1(x,t)$: Total power consumption of PMs at instant t;
- $pmax_i$: Maximum power consumption of a PM H_i;
- $pmin_i$: Minimum power consumption of a PM H_i; As suggested in [8], $pmin_i \approx pmax_i * 0.6$;
- $Ur_{1,i}(t)$: Utilization ratio of resource 1 (in this case CPU) by H_i at instant t;
- $Y_i(t) \in \{0,1\}$: Indicates if H_i is turned on $(Y_i(t) = 1)$ or not $(Y_i(t) = 0)$ at instant t.

2.4.2 Economical Revenue Maximization

Equation (8) represents leasing costs, defined as the sum of the total costs of leasing each VM V_j that is effectively allocated for execution on any PM of an alternative datacenter of the cloud federation. A provider must offer its idle resources to the cloud federation at lower prices than offered to customers in the actual cloud market for the federation to make sense. The pricing scheme may depend on the particular agreement between providers of the cloud federation [9]. For simplicity, this formulations considers that the main provider may lease requested resources (that are not able to provide) from the cloud federation at 70% $(\hat{X}_j = 0.7)$ of its market price $(R_j(t))$. These Leasing Costs $(LC(t))$ may be formulated as:

$$LC(t) = \sum_{j=1}^{m(t)} (R_j(t) \times X_j(t) \times \hat{X}_j) \quad (8)$$

where:

- $LC(t)$: Total leasing costs at instant t;
- $R_j(t)$: Economical revenue for attending V_j in [$] at instant t;
- $X_j(t) \in \{0,1\}$: Indicates if V_j is allocated for execution on a PM $(X_j(t) = 1)$ or not $(X_j(t) = 0)$ at instant t;
- $\hat{X}_j$: Indicates if V_j is allocated on the main provider $(\hat{X}_j = 0)$ or on an alternative datacenter of the cloud federation $(\hat{X}_j = 0.7)$;
- $m(t)$: Number of VMs at each discrete time t, where $1 \leq m(t) \leq m_{max}$.

It is important to note that $\hat{X}_j$ is not necessarily a function of time. The decision of locating a VM V_j on a federated provider is considered only in the placement process, with no possible migrations between different IaaS providers.

Additionally, overbooked resources may incur in unsatisfied demand of resources at some periods of time, causing Quality of Service (QoS) degradation, and consequently Service Level Agreement (SLA) violations with economical penalties. These economical penalties should be minimized for an economical revenue maximization. Based on the workload independent QoS metric presented in [8], formalized in SLAs, Equation (9) represents total economical penalties for SLA violations, defined as the sum of the total penalties costs for unsatisfied demand of resources.

$$EP(t) = \sum_{j=1}^{m(t)} \left(\sum_{k=1}^{r} Rr_{k,j}(t) \times \Delta r_{k,j}(t) \times X_j(t) \times \phi_k \right) \quad (9)$$

where:

- $EP(t)$: Total economical penalties at instant t;
- r: Number of considered resources. In this paper 3: CPU, RAM memory and network capacity;

$Rr_{k,j}(t)$: Economical revenue for attending $Vr_{k,j}(t)$;
$\Delta r_{k,j}(t)$: Ratio of unsatisfied resource k at instant t where $\Delta r_{k,j}(t) = 1$ means no unsatisfied resource, while $\Delta r_{k,j}(t) = 0$ means resource k is unsatisfied in 100%;
$X_j(t) \in \{0,1\}$: Indicates if V_j is allocated for execution on a PM ($X_j(t) = 1$) or not ($X_j(t) = 0$) at instant t;
ϕ_k: Penalty factor for resource k, where $\phi_k \geq 1$;
$m(t)$: Number of VMs at each discrete time t, where $1 \leq m(t) \leq m_{max}$.

In this work, the maximization of the total economical revenue that an IaaS provider receives is achieved by minimizing the total costs of leasing resources from alternative datacenters of the cloud federation as well as the total economical penalties for SLA violations, as presented in (10), i.e.

$$f_2(x,t) = LC(t) + EP(t) \tag{10}$$

where:

$f_2(x,t)$: Total economical expediture of the main IaaS provider at instant t.

2.4.3 Resources Utilization Maximization

This work considers a maximization of the resource utilization by minimizing the average ratio of wasted resources on each PM H_i (i.e. resources that are not allocated to any VM V_j).

$$f_3(x,t) = \frac{\sum_{i=1}^{n} \left[1 - \left(\frac{\sum_{k=1}^{r} Ur_{k,i}(t)}{r} \right) \right] \times Y_i(t)}{\sum_{i=1}^{n} Y_i(t)} \tag{11}$$

where:

$f_3(x,t)$: Average ratio of wasted resources at instant t;
$Ur_{k,i}(t)$: Utilization ratio of resource k of PM H_i at instant t;
r: Number of considered resources. In this paper $r = 3$: CPU, RAM memory and network capacity.

2.4.4 Reconfiguration Time Minimization

Inspired in [10], once a placement reconfiguration is accepted in the VMPr phase, all VM migrations are assumed to be performed in parallel through a management network exclusively used for these actions, increasing 10% CPU utilization in VMs being migrated. Consequently, the minimization of the (maximum) reconfiguration time could be achieved by minimizing the maximum amount of memory to be migrated from one PM H_i to another $H_{i'}$ $(i \neq i')$.

Equation (12) was proposed in [1] to minimize the maximum amount of RAM memory that must be moved between PMs at instant t.

$$f_4(x,t) = \max(MT_{i,i'}) \quad \forall i, i' \in \{1,\ldots,n\} \tag{12}$$

where:

$f_4(x,t)$: Network traffic overhead for VM migrations at instant t;
$MT_{i,i'}$: Total amount of RAM memory to be migrated from PM H_i to $H_{i'}$.

The following sub-section summarizes the main considerations taken into account to combine the four presented objective functions into a single objective function to be minimized with the aim of having a single figure of merit (or optimization metric).

2.5 Normalization and Scalarization

Each considered objective function must be formulated in a single optimization context (in this case, minimization) and each objective function cost must be normalized to be comparable and combinable as a single objective. This work normalizes each objective function cost by calculating $\hat{f}_i(x,t) \in \mathbb{R}$, where $0 \leq \hat{f}_i(x,t) \leq 1$ for each objective function $f_i(x,t)$.

$$\hat{f}_i(x,t) = \frac{f_i(x,t) - f_i(x,t)_{min}}{f_i(x,t)_{max} - f_i(x,t)_{min}} \tag{13}$$

where:

$\hat{f}_i(x,t)$: Normalized cost of objective function $f_i(x,t)$ at instant t;
$f_i(x,t)$: Cost of original objective function $f_i(x,t)$;
$f_i(x,t)_{min}$: Minimum possible cost for $f_i(x,t)$;
$f_i(x,t)_{max}$: Maximum possible cost for $f_i(x,t)$.

The presented normalized objective functions are combined into a single objective considering a minimum Euclidean distance to the origin, expressed as:

$$F(x,t) = \sqrt{\sum_{i=1}^{q} \hat{f}_i(x,t)^2} \tag{14}$$

where:

$F(x,t)$: Single objective function combining each $\hat{f}_i(x,t)$ at instant t;
$\hat{f}_i(x,t)$: Normalized cost of objective function $f_i(x,t)$ at instant t;
q: Number of objective functions.

2.6 Scenario-based Uncertainty Modeling

In this work, uncertainty is modeled through a finite set of well-defined scenarios S [11], where the following uncertain parameters are considered: (i) virtual resources capacities (vertical elasticity), (ii) number of VMs that compose cloud services (horizontal elasticity), (iii) utilization of CPU and RAM memory virtual resources and (iv) utilization of networking virtual resources (both relevant for overbooking).

For each scenario $s \in S$, a temporal average value of the objective function $F(x,t)$ presented in (14) is calculated as:

$$\overline{f_s(x,t)} = \frac{\sum_{t=1}^{t_{max}} F(x,t)}{t_{max}} \quad (15)$$

where:

$\overline{f_s(x,t)}$: Temporal average of combined objective function for all discrete time instants t in scenario $s \in S$;

t_{max}: Duration of a scenario in discrete time instants.

As previously described, when parameters are uncertain, it is important to find solutions that are acceptable for any (or most) considered scenario $s \in S$. This work considers minimization of the average objective function costs criteria [11] to select among solutions:

$$F_1 = \overline{F(x,t)} = \frac{\sum_{s=1}^{|S|} \overline{f_s(x,t)}}{|S|} \quad (16)$$

where:

F_1: Average $\overline{f_s(x,t)}$ for all scenarios $s \in S$ [11].

3 Evaluated Algorithms

Considering a previous research work of some of the authors [1], promising results of the proposed algorithm were found in order to implement it in real-world IaaS middlewares. The mentioned proposed algorithm considers a two-phase optimization scheme using *First-Fit Decreasing* (FFD) for the iVMP phase, a *Memetic Algorithm* (MA) for the VMPr phase, a prediction-based method for VMPr Triggering and an update-based method for VMPr Recovering. This algorithm was denoted as *Algorithm 3 (A3)* in [1] and is considered in this work as *Algorithm 1 (A1)* for the presented experimental evaluation.

Additionally, and as a first step on implementing *A1* in a real-world IaaS middleware, official OpenStack algorithms for VMP were studied [12]. In this context, two alternatives are available for configuring VMP processes in OpenStack: (i) *Filter Scheduler* and (ii) *Random Scheduler*. Taking into account that the *Random Scheduler* uses a trivial logic for solving the VMP, this work considers the *Filter Scheduler* as *Algorithm 2 (A2)* for the presented experimental evaluation.
It is important to note that *A2* considers only the iVMP phase for its operation, without taking into account migration of VMs between PMs.

The following sub-sections briefly present some relevant aspects on evaluated algorithms *A1* and *A2*.

3.1 Algorithm 1: Two-Phase Optimization

This section presents details on algorithm *A1* [1] as considered iVMP and VMPr algorithms as well as considered VMPr Triggering and Recovering methods.

3.1.1 Incremental VMP (iVMP) for A1

In experimental results previously obtained by some of the authors in [2], the First-Fit Decreasing (FFD) heuristic outperformed other evaluated heuristics in average; consequently, the mentioned heuristic was considered in *A1* for the iVMP phase (see Table 1). In the First-Fit (FF) heuristic, requested VMs $V_j(t)$ are allocated on the first PM H_i with available resources. The considered FFD heuristic operates similarly to FF heuristic, with the main difference that FFD heuristic sorts the list of requested VMs $V_j(t)$ in decreasing order by revenue $R_j(t)$ (see details in Algorithm 1).

Taking into account the particularities of the proposed complex IaaS environment, the FFD heuristic presents some modifications when comparing to the one presented in [2], mainly considering the cloud service request types previously described in Section 2.1. In fact, Algorithm 1 shows that cloud service destruction, scale-down of VM resources and cloud services scale-in are processed first, in order to release resources for immediate re-utilization (steps 1-3 of Algorithm 1). At step 4, requests from $V(t)$ are sorted by a given criterion as revenue ($R_j(t)$) in decreasing order (of course, other criterion may be considered, as CPU [2]), where scale-up of VM resources and cloud services scale-out are firstly processed (steps 5-6), in order to consider elastic cloud services more important than non-elastic ones. Next, unprocessed requests from $V_j(t)$ include only cloud service creations that are allocated in decreasing order (steps 7-18). Here, a V_j is allocated in the first H_i with available resources after considering previously sorted $V(t)$. If no H_i has sufficient resources to host V_j, it is allocated in another federated provider. Finally, the placement $x(t+1)$ is updated and returned (steps 19-20).

3.1.2 VMP Reconfiguration (VMPr) for A1

Previous research work by the authors focused on developing VMPr algorithms considering centralized decisions such as the offline MAs presented in [13, 5, 6]. In this work, the considered VMPr algorithm for *A1* is based on the one presented in [5] and it works in the following way (see details in Algorithm 2):

At step 1, a set Pop_0 of candidate solutions is randomly generated. These candidate solutions are repaired at step 2 to ensure that Pop_0 contains only feasible solutions, satisfying defined constraints.

Then, the algorithm tries to improve candidate solutions at step 3 using local search. With the obtained solutions, elitism is applied and the first best solution $x'(t)$ is selected from $Pop''_0 \cup x(t)$ at step 4 using objective function defined in (14). After an initialization in step 5, evolution begins (steps 6-12). The evolutionary process basically follows a similar behavior: solutions are selected from the union of the evolutionary set of solutions (or population), also known as Pop_u, and the best known solution $x'(t)$ (step 7), crossover and mutation operators are applied as usual (step 8), and

Algorithm 1: First-Fit Decreasing (FFD) for iVMP phase in Algorithm $A1$.

Data: $H, V(t), U(t), x(t)$ (see notation in Section 2.1)
Result: Incremental Placement $x(t+1)$
process cloud services destruction from $V(t)$;
process scale-down of VMs resources from $V(t)$;
process cloud services scale-in from $V(t)$;
sort VMs by revenue ($R_j(t)$) in decreasing order;
process scale-up of VMs resources from $V(t)$;
process cloud services scale-out from $V(t)$;
foreach *unprocessed* V_j *in* $V(t)$ **do**
  while V_j *is not allocated* **do**
    foreach H_i *in* H **do**
      if H_i *has enough resources to host* V_j **then**
        allocate V_j into H_i and ***break*** loop;
      end if
    end foreach
    if V_j *is still not allocated* **then**
      allocate V_j in another federated provider;
    end if
  end while
end foreach
update $x(t+1)$ with processed requests;
return $x(t+1)$

Algorithm 2: Memetic Algorithm (MA) for VMPr phase in Algorithm $A1$.

Data: $H, U(t), x(t)$ (see notation in Section 2.1)
Result: Recalculated Placement $x'(t)$
initialize set of candidate solutions Pop_0;
Pop'_0 = repair infeasible solutions of Pop_0;
Pop''_0 = apply local search to solutions of Pop'_0;
$x'(t)$ = select best solution from $Pop''_0 \cup x(t)$ considering (14);
$u = 0; Pop_u = Pop''_0$;
while *stopping criterion is not satisfied* **do**
  Pop_u = selection of solutions from $Pop_u \cup x'(t)$;
  Pop'_u = crossover and mutation on solutions of Pop_u;
  Pop''_u = repair infeasible solutions of Pop'_u;
  Pop'''_u = apply local search to solutions of Pop''_u;
  $x'(t)$ = select best solution from Pop'''_u considering (14);
  increment number of generations u;
end while
return $x'(t)$

eventually solutions are repaired, as there may be infeasible solutions (step 9). Improvements of solutions of the evolutionary population Pop_u may be generated at step 10 using local search (local optimization). At step 11, the best known solution $x'(t)$ is updated (if applicable), while at step 12 the generation (or iteration) counter is updated. The evolutionary process is repeated until the algorithm meets a stopping criterion, returning the best known solution $x'(t)$ for a placement reconfiguration. More details may be found in [5].

Algorithm 3: Update-based VMPr Recovering in Algorithm $A1$.

Data: $x(t), x'(t-\beta)$ (see notation in Section 2.1)
Result: Recovered Placement $x'(t)$
remove VMs V_j from $x'(t-\beta)$ that are no longer running in $x(t)$
adjust resources from $x'(t-\beta)$ that changed in $x(t)$
add VMs V_j from $x(t)$ that were not considered in $x'(t-\beta)$
if $x'(t-\beta)$ *is better than* $x(t)$ **then** ;
  return $x'(t-\beta)$;
else return $x(t)$;

3.1.3 Prediction-based Triggering for A1

In this work, *A1* considers a prediction-based method that analyses objective function (see (14)), in a way that it is possible to detect situations where a placement might be required for reconfiguration purposes.

The presented prediction-based VMPr Triggering method considers *Double Exponential Smoothing* (DES) [14] as a statistical technique for predicting values of the objective function $F(x,t)$, as formulated next in (17) to (19):

$$S_t = \alpha \times Z_t + (1-\tau)(S_{t-1} + b_{t-1}) \quad (17)$$

$$b_t = \tau(S_t - S_{t-1}) + (1-\tau)(b_{t-1}) \quad (18)$$

$$\overline{Z}_{t+1} = S_t + b_t \quad (19)$$

where:
α: Smoothing factor, where $0 \leq \alpha \leq 1$;
τ: Trend factor, where $0 \leq \tau \leq 1$;
Z_t: Known value of $F(x,t)$ at discrete time t;
S_t: Expected value of $F(x,t)$ at discrete time t;
b_t: Trend of $F(x,t)$ at discrete time t;
$\overline{Z}_{t+1}$: Value of $F(x,t+1)$ predicted at discrete time t.

At each discrete time t, the VMPr Triggering method predicts next N values of $F(x,t)$ and triggers the VMPr phase in case $F(x,t)$ is predicted to consistently increase, considering that $F(x,t)$ is minimized.

3.1.4 Update-based Recovering for A1

When considering a two-phase optimization scheme for the VMP problem in cloud computing environments, the placement reconfiguration obtained in the VMPr phase is regarded as obsolete as time progresses during the algorithm running time due to its offline nature. That is why a new way of improving the placement taking into account the new requests is needed. The iVMP phase performs the recalculation of the improved placement. Consequently, the calculated new placement must be recovered according to the considered VMPr Recovering method before the reconfiguration is performed in operations.

The considered update-based VMPr Recovering method receives the placement reconfiguration calculated in the VMPr phase (corresponding to the discrete

time $t-\beta$) and the current placement $x(t)$ as input data, as summarized in Algorithm 3.

Considering that any VM V_j could be destroyed, or a cloud service could be scaled-in (horizontal elasticity) during the β discrete times where the calculation of the placement reconfiguration was performed, these destroyed VMs are removed from $x'(t-\beta)$ (step 1). Next, any resource from a VM V_j could be adjusted due to a scale-up or scale-down (vertical elasticity). Consequently, these resource adjustments are performed in $x'(t-\beta)$ (see step 2). Additionally, new VMs V_j could be created, or a cloud service could be scaled-out (horizontal elasticity), during the calculation of $x'(t-\beta)$. Finally, if the partially recalculated placement $x'(t-\beta)$ is better than the current placement $x(t)$, $x'(t-\beta)$ is accepted (step 5) and the corresponding management actions are performed (i.e. mainly migration of VMs between PMs). In case $x'(t-\beta)$ is not better than the current placement $x(t)$, no change is performed and the VMPr phase finishes without any further consequence.

3.2 Algorithm 2: Filter Scheduler

This work also evaluates the current default OpenStack Scheduler [12] for allocating VMs into PMs, identified as *A2*. This OpenStack VMP algorithm (*A2*) considers *filtering* and *weighting* for selecting a PM H_i to host a requested VM V_j for the considered iVMP phase.

For each requested VM V_j, the following set of filters are firstly applied to determine which PMs are eligible for allocating each requested VM:

- *RetryFilter*: if the PM H_i is available to host VMs. This is considered in the uncertain formulation with the binary variable $Y_i(t)$ that indicates if H_i is turned on ($Y_i(t)=1$) or not ($Y_i(t)=0$).
- *AvailabilityZoneFilter*: if the PM H_i is in the requested availability zone. The availability zone is mapped as a datacenter identifier c_i to fit in the considered uncertain formulation.
- *ComputeFilter*, *RamFilter*, *DiskFilter*: if the PM H_i has sufficient computational resources for allocating requested VM, as input data on $V(t)$.
- *ComputeCapabilitiesFilter*: to ensure satisfaction of additional specifications associated with the requested VM image. This is not considered in the uncertain formulation.
- *ImagePropertiesFilter*: to ensure that PM H_i has properties specified on the VM image. This is not considered in the uncertain formulation.
- *ServerGroupAntiAffinityFilter*: (if requested) to ensure that the requested VM will be allocated in a different PM than other VMs that compose the cloud service S_b.

Next, pre-selected PMs considering applied filters are then processed and weights are assigned to each PM, based on VM request specifications. Finally, PMs with the highest weight is selected and an incremental placement for the next time instant is returned. Table 1 summarize evaluated algorithms and methods.

Algorithm 4: Filter Scheduler in Algorithm *A2*.

```
Data: H, V(t), U(t), x(t) (see notation in Section 2.1)
Result: Incremental Placement x(t+1)
foreach V_j in V(t) do
    filtered − PMs = list of suitable PMs by applying
      filtering criteria
end foreach
foreach V_j in V(t) do
    weighted − PMs = weight PMs from
      filtered − PMs
    select PM with the highest weight
end foreach
return Incremental Placement x(t+1)
```

4 Experimental Evaluation

The following sub-sections summarize the experimental environment as well as the main findings identified in the experiments performed as part of this work to validate the *Algorithm (A1)* proposed in [1] against the OpenStack Filter Scheduler, *Algorithm A2* (see Table 1), considering scenario-based simulations with 400 different scenarios, taking into account average objective functions costs (see (16)).

4.1 Experimental Environment

The evaluated algorithms were implemented using Java programming language and considering the *Dynamic VMP Framework* available online[2]. Experiments were performed on a Windows 10 Operating System with an AMD A8-7410 APU with AMD Radium Graphics at 2.2 GHz CPU and 8 GB of RAM.

For more details on the considered experimental environment, as well as the 400 designed experimental workloads, interested readers may refer to [1].

4.2 Experimental Results

The main goal of the presented experimental evaluation is to validate that the previously proposed Algorithm *A1* [1] may result in a competitive implementation on an IaaS middleware such as OpenStack.

Table 2 presents values of the considered evaluation criteria, i.e. F_1 costs (see (16)), summarizing results obtained in performed simulations. The mentioned evaluation criteria are presented separately for each of the five considered IaaS cloud datacenter. It is worth noting that the considered IaaS cloud datacenters represent datacenters of different sizes and consequently, the considered workload traces represent different load of requested CPU resources (e.g. Low ($\leq 30\%$), Medium ($\leq 60\%$), High ($\leq 90\%$), Full ($\leq 98\%$) and Saturate ($\leq 120\%$)) workloads.

[2] http://github.com/DynamicVMP/dynamic-vmp-framework/releases

Table 1: Summary of evaluated algorithms as well as their corresponding VMPr Triggering and Recovering methods. N/A indicates a Not Applicable criterion.

Characteristics / Algorithm	**Decision**	**iVMP**	**VMPr**	**VMPr Triggering**	**VMPr Recovering**
A1 - inspired in [1]	Centralized	FFD	MA	Prediction-based	Update-based
A2 - inspired in [12]	N/A	Filter Scheduler	N/A	N/A	N/A

Table 2: Summary of evaluation criteria in experimental results for evaluated algorithms.

Criterion	Algorithm	Datacenter					
		DC_1	DC_2	DC_3	DC_4	DC_5	Ranking
F_1	**A1**	**0.752**	**0.838**	**0.926**	**0.934**	**0.983**	1^{st}
	A2	0.794	0.932	0.986	1.003	1.019	2^{nd}

Based on the information presented in Table 2, it can be seen that Algorithm $A1$ outperformed Algorithm $A2$ in every experiment, taking into account the considered evaluation criterion (F_1). In summary, Algorithm $A1$ obtained better results (minimum cost) for considered evaluation criterion. When considering average objective function costs (F_1) as evaluation criterion, Algorithm $A1$ obtained between 4% and 11% better results than Algorithm $A2$.

5 Conclusions and Future Work

This work performed a first experimental evaluation of a previously proposed [1] two-phase optimization scheme for VMP problems in complex cloud computing environments, towards its implementation in a real-world IaaS middleware. For this, an industry de-facto standard as *OpenStack* was chosen and the *Filter Scheduler* was slightly adapted for simulations taking into account the considered VMP formulation.

The experimental evaluation presented in this work was mainly guided by previous work by some of the authors, considering that main contributions firstly proposed in [1] were taken into account to compare most promising studied algorithms ($A1$ in this case) against algorithms inspired in real-world ones (i.e. $A2$).

Experimental results demonstrate that the proposed algorithm $A1$ outperformed $A2$ in all considered experiments and may be considered as a promising algorithm for its implementation. Even do, several challenges still need to be faced in order address a good proposed tools for cloud computing datacenter management.

As a first step, IaaS middlewares such as *OpenNebula*, *vSphere Cloud* and other alternative tools with VMP algorithms should still be evaluated against Algorithm $A1$. This is proposed as future work.

Additionally, several assumptions should still be adapted to real-world situations or at least be evaluated under more scenarios, such as the recalculation time β that until now has been assumed to be a constant of discrete time instants. In real-world operations, this should be considered as a function of time t.

Several future works were also identified, mainly considering the novelty of the considered formulation. First, a formulation of a VMP problem considering a dynamic set of PMs $H(t)$, to consider PM crashes, maintenance or even deployment of new generation hardware is proposed as a future work.

Although modeling power consumption considering a linear relationship with CPU utilization is a very accepted approach in the specialized literature, considering the impact of other resources such as RAM and networking is proposed as future work.

Considering VMP formulations with more sophisticated cloud federation approaches is also left as a future work, taking into account the basic cloud federation approach considered in this work. Additionally, an experimental evaluation of alternative algorithms for both iVMP and VMPr phase is proposed as a future work, in order to explore performance issues with the proposed VMPr Triggering and Recovering methods.

Novel VMPr Triggering and VMPr Recovering methods could still be proposed to improve the considered two-phase optimization scheme in $A1$. The authors of this work also recognized the importance of jointly considering auto-scaling algorithms with the proposed two-phase optimization scheme for VMP problems, mainly for elastic cloud services as the considered in this work.

Experimenting with geo-distributed datacenters is also left as a future work, taking into account that simulations presented in this work considered only one cloud computing datacenter. Finally, fixed pricing is still very popular in cloud computing markets but emerging pricing schemes such as Spot Prices [15] should also be considered in real-world cloud computing datacenter operations.

6 Acknowledgements

This research is currently supported by CONACYT, in the context of the PINV15-781 "Software-defined Datacenters" research project grant.

References

[1] F. López-Pires, B. Barán, L. Benítez, S. Zalimben, and A. Amarilla, "Virtual machine placement for elastic infrastructures in overbooked cloud computing datacenters under uncertainty," *Future Generation Computer Systems*, vol. 79, pp. 830 – 848, 2018.

[2] F. López-Pires, B. Barán, A. Amarilla, L. Benítez, R. Ferreira, and S. Zalimben, "An experimental comparison of algorithms for virtual machine placement considering many objectives," in *9th Latin America Networking Conference (LANC)*, pp. 75–79, 2016.

[3] Z. Á. Mann, "Allocation of virtual machines in cloud data centers - A survey of problem models and optimization algorithms," *ACM Computing Surveys (CSUR)*, vol. 48, no. 1, p. 11, 2015.

[4] F. López-Pires and B. Barán, "A virtual machine placement taxonomy," in *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*, pp. 159–168, IEEE Computer Society, May 2015.

[5] D. Ihara, F. López-Pires, and B. Barán, "Many-objective virtual machine placement for dynamic environments," in *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, pp. 75–79, IEEE, 2015.

[6] F. López-Pires and B. Barán, "A many-objective optimization framework for virtualized datacenters," in *Proceedings of the 2015 5th International Conference on Cloud Computing and Service Science*, pp. 439–450, 2015.

[7] F. López-Pires and B. Barán, "Virtual machine placement literature review," *http://arxiv.org/abs/1506.01509*, 2015.

[8] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.

[9] M. Gahlawat and P. Sharma, "Survey of virtual machine placement in federated clouds," in *Advance Computing Conference (IACC), 2014 IEEE International*, pp. 735–738, Feb 2014.

[10] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[11] M. A. Aloulou and F. Della Croce, "Complexity of single machine scheduling problems under scenario-based uncertainty," *Operations Research Letters*, vol. 36, no. 3, pp. 338–342, 2008.

[12] D. OpenStack, "Scheduling." `https://docs.openstack.org/mitaka/config-reference/compute/scheduler.html`, 2018. [Online; accessed 14-June-2018].

[13] F. López-Pires and B. Barán, "Multi-objective virtual machine placement with service level agreement: A memetic algorithm approach," in *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, pp. 203–210, IEEE Computer Society, 2013.

[14] J. Huang, C. Li, and J. Yu, "Resource prediction based on double exponential smoothing in cloud computing," in *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, pp. 2056–2060, April 2012.

[15] S. Arévalos, F. López-Pires, and B. Barán, "A comparative evaluation of algorithms for auction-based cloud pricing prediction," in *2016 IEEE International Conference on Cloud Engineering (IC2E)*, pp. 99–108, April 2016.

www.jcc.info.unlp.edu.ar
jcc@lidi.info.unlp.edu.ar

POSTGRADO
FACULTAD DE INFORMÁTICA
UNLP

III - LIDI Instituto de Investigación en Informática - LIDI

RedUNCI