ARTÍCULO BREVE DE INVESTIGACIÓN

# Implementación de un cluster de computadoras

**Dan Álvarez[2], Jeancarlo Barrios1, Kuk Ho Chung[2], Rafael León[1], Alberto López[2], Alberto Suriano[1], Alejandro Vásquez[1], Luis R. Furlan[3] y José Tomás Prieto[3,4,5]**

[1]Departamento de Ingeniería en Ciencias de la Computación y Tecnologías de la Información, [2]Departamento de Ingeniería Mecatrónica, [3]Centro de Estudios en Informática Aplicada, [4]Centro de Estudios en Salud, Instituto de Investigaciones, Universidad Del Valle de Guatemala, [5]CRG, École Polytechnique, Francia
sur12281@uvg.edu.gt

**RESUMEN:** El objetivo de esta investigación es describir la creación e implementación de un *cluster* de computadoras dentro del campus central de la Universidad del Valle de Guatemala, además de proporcionar una demostración que ayude a clarificar los beneficios de este tipo de herramientas. El uso de *cluster* de computadoras busca hacer más rápida la ejecución de algoritmos que son computacionalmente pesados, además de hacer más eficiente el uso de recursos para la solución de problemas. Se usó la herramienta *open-source BOINC* para poder desarrollar un *cluster*, el cual, tras haber sido evaluado en una red local, se implementó en la nube en *AWS* para poder ser accedido de forma remota. Al usar el *cluster* en la ejecución del algoritmo para buscar una solución del problema de matrices de Hadamard, se notó que la ejecución del mismo fue más rápida en un 67 %, comparada con la ejecución en una sola máquina. La importancia de este trabajo radica en dejar un planteamiento claro de cómo se puede desarrollar este tipo de herramientas, en clarificar los beneficios de la misma, y en sentar las bases para el futuro de nuestra investigación.

**PALABRAS CLAVE:** Cluster, *BOINC*, computación distribuida.

## Implementation of a computer cluster

**ABSTRACT:** The goal of the present study is to describe the creation and implementation of a computer cluster in the central campus of the Universidad del Valle de Guatemala, and to provide a test demonstration of the benefits of this kind of computational tool. The use of a computer cluster is an attempt to improve the execution time of computationally heavy algorithms, and to make a more efficient use of computational resources in solving a problem. The open-source tool *BOINC* was used to develop a cluster, which, after being evaluated in a local network, was implemented in the *AWS* cloud to be accessed remotely. After using the cluster in the execution of the algorithm to find the solution to the problem of the Hadamard matrixes, its execution time was 67 % faster, in comparison with the execution in only one computer. This work's importance lies in leaving a clear approach to how to develop this kind of tools, clarifying its benefits, and lay the groundwork for future investigations.

**KEYWORDS:** Cluster, *BOINC*, distributed computing.

## Introducción

En el campo de la computación existen problemas de alto costo en términos de procesamiento y tiempo. Por ello, es necesario utilizar técnicas diversas que buscan mejorar la eficiencia y agilizar procesos para que sea factible resolver problemas que de otra forma tomarían demasiado tiempo o bien resultarían imposibles de resolver computacionalmente. Una de las formas de lograr esto es distribuir la tarea en diferentes computadoras, llamadas nodos, controladas por una computadora principal, o nodo maestro, constituyendo así un *cluster* de computadoras[1].

[1] Ver Anexo 1. Glosario de términos.

En informática, se denomina *cluster* a un conjunto de computadoras interconectadas por medio de un hardware de red de alta velocidad con el objetivo de sumar las capacidades individuales de cada computadora y procesar con mayor eficiencia grandes cantidades de datos, ofreciendo un bajo costo y gran escalabilidad (Milone, 2002:173). En un *cluster* de computadoras, los múltiples procesadores usualmente trabajan en paralelo, y cada procesador tiene más de un núcleo computacional. Por lo tanto, el nivel de paralelismo de un *cluster* es mucho más alto que el de un procesador multi-núcleo (Tao, Zhang y Laili, 2015:132).

Según Hwang et al. (2012), el diseño objetivo de un *cluster* de computadoras se clasifica mediante seis atributos: escalabilidad, compartimiento, control, homogeneidad, programabilidad y seguridad.

Se entiende como escalabilidad la habilidad de adaptación que un sistema posee al enfrentarse a un crecimiento continuo de trabajo sin sacrificar la calidad y eficiencia en el servicio que proporciona. Existen dos tipos de escalabilidad: vertical y horizontal. Un *cluster* tiene escalabilidad vertical cuando se mejora la capacidad del hardware de alguna de las computadoras. Un *cluster* es escalable horizontalmente cuando se añaden más computadoras al sistema (Cesarini y Vinoski, 2016:405).

El *packaging* o empaquetado se refiere a la forma en que los nodos de un *cluster* están disponibles, es decir, pueden estar en un *cluster* compacto donde los nodos están en un solo lugar, o en un *cluster* suelto, donde los nodos pueden estar en diferentes lugares, edificios o regiones remotas, y por lo general se conectan por medio de una red de área local (*LAN*, por sus siglas en inglés) o una red de área amplia (*WAN*, por sus siglas en inglés), estas se definen como redes de comunicación entre computadoras a diferentes escalas. *LAN* por su parte se refiere a las redes que se conectan a través de un dispositivo en un área pequeña como una residencia. Mientras que *WAN* se refiere a redes que conectan dispositivos a grandes distancias, por ejemplo: el internet (Donahue, 2007).

Por otro lado, el término control se refiere al control centralizado o descentralizado del *cluster*. Un *cluster* compacto normalmente es controlado de forma centralizada, mientras que un *cluster* suelto puede ser controlado de cualquier forma. Se dice que un *cluster* es homogéneo cuando se utilizan nodos provenientes de una misma plataforma, arquitectura de procesador y el mismo sistema operativo. En un *cluster* heterogéneo todos los nodos poseen arquitecturas diferentes. La seguridad de un *cluster* puede ser expuesta o cerrada. En un *cluster* expuesto la comunicación entre los nodos puede ser accedida por el mundo exterior con protocolos como *TCP/IP*. En un *cluster* cerrado, la comunicación está protegida del mundo exterior y se utiliza en *clusters* comerciales o de investigaciones (Hwang et al. 2012:68-70).

Entre las herramientas más populares para la realización de *clusters* de computadoras, se encuentran *BOINC* y *Rocks*. *BOINC* es una de las herramientas más prometedoras y permite crear supercomputadoras virtuales *(VCSC); i.e.* supercomputadoras conformadas por un conjunto de computadoras normales (desktops, laptops, etc.), todas conectadas a través de una red común. La página web de *BOINC* asevera que, si se tienen 10,000 equipos conectados al *VCSC*, y el tiempo de operación de cada uno es del 50%, los costos de mantenimiento del cluster se reducen de 1 millón de dólares a 10,000 dólares (*BOINC*, 2008). Existen dos tipos de *cluster* que se pueden crear con *BOINC: clusters* que utilizan computadoras voluntarias (*Volunteer computing*) y *clusters* de mallas computacionales (*Grid computing*). Los primeros utilizan computadoras que pertenecen a individuos que prestan la capacidad de procesamiento de su computadora en tiempos de desuso, mientras que el *cluster* de malla computacional utiliza computadoras cuyo único fin es formar parte de la supercomputadora (BOINC, 2008).

Por otro lado, en lo referente a *Rocks* (originalmente llamado *NPACI Rocks*), se puede decir que es un conjunto de herramientas de fuente abierta para construir *clusters* de alto rendimiento. El objetivo principal de *Rocks* es hacer que los procedimientos de instalación de un *cluster* sean lo más fácil posible. Al instalar *Rocks,* también se instala el *clustering software* y una versión actualizada de *Red Hat Linux;* no se puede agregar *Rocks* a un servidor existente o usar éste con alguna otra distribución de *Linux,* aunque las últimas versiones, lanzadas a partir de 2008, están basadas en *CentOS* (una distribución de *Red Hat* pero de clase empresarial), con un ambiente gráfico que facilita la instalación en todos los nodos del *cluster*. La instalación por defecto tiende a ser muy rápida y dependiendo del hardware puede ser posible reinstalar un nodo en diez minutos o menos (Sloan, 2005:121).

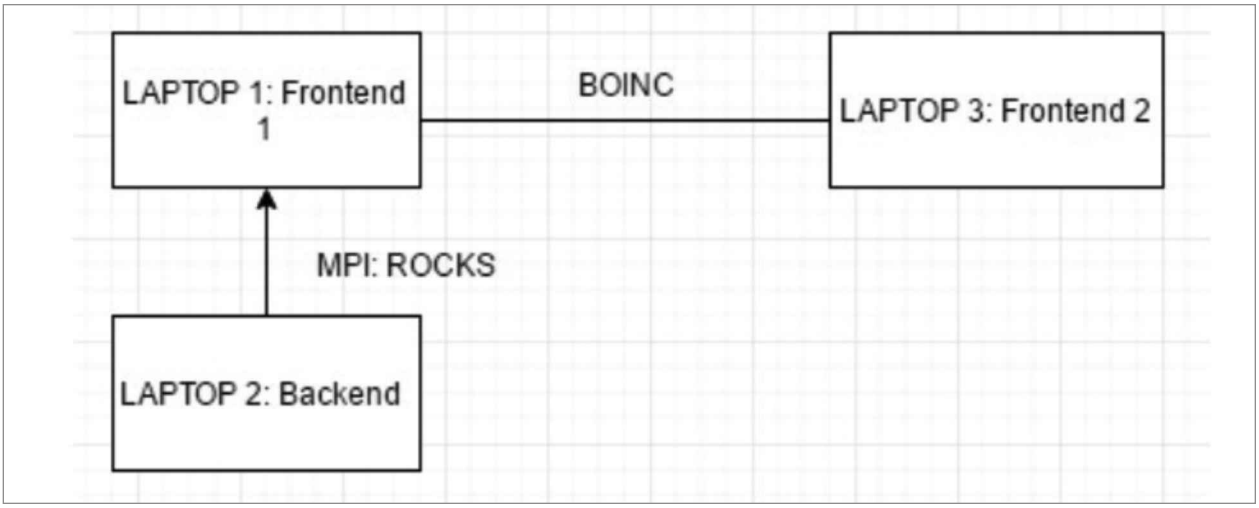
## Materiales y métodos

La implementación y evaluación del *cluster* de computadoras consistió en cuatro fases que se presentan a continuación.

### Fase 1: Investigación sobre herramientas disponibles

Como fase inicial, se investigaron diferentes herramientas que podrían servir para implementar el *cluster*. Se consideraron herramientas como *BOINC, Eucalyptus, MPI* y *Rocks*. De éstas, se eligieron *BOINC* y *Rocks* para realizar pruebas con las cuales se decidiría cuál de las dos herramientas se usaría en el proyecto. Estas fueron elegidas de acuerdo a las recomendaciones encontradas en las páginas oficiales de *BOINC* y *Rocks*, además de las facilidades que tenían cuando se investigó sobre ellas.

Se consultó directamente con empleados de *BOINC* vía correo electrónico para obtener una opinión experta acerca del sistema y sus aplicaciones para el *cluster* planeado en este proyecto. Según David Anderson (2016), este sistema no es adecuado para la creación de un *cluster*, pues inicialmente fue diseñado para interconectar dispositivos con características de hardware y software variadas (por ejemplo, interconectar laptops y desktops con distintos sistemas operativos). Adicionalmente, *BOINC* no cuenta con un sistema de manejo de archivos integrado. Puesto




**Gráfica 1.** Distribución de Computadoras en Prueba de *Rocks*.

que un *cluster* cuenta con un sistema de almacenamiento compartido, y las computadoras utilizadas son todas similares, David recomendó buscar otras opciones creadas específicamente para *clusters*, como *Slurm, Rocks* o *Condor*.

## BOINC vs Rocks

En cuanto a *Rocks*, se realizaron dos tipos de pruebas, una utilizando máquinas virtuales y la otra haciendo uso de máquinas físicas. Para el primer caso se decidió interconectar dos computadoras utilizando máquinas virtuales. Se utilizó una computadora con procesador AMD A6-5200 de cuatro núcleos a 2 GHz, y 4 GB de RAM. Se instaló *VirtualBox*, y se crearon dos máquinas virtuales dentro de este programa. Una de ellas representaba el *master* o *frontend*, que controlaba a la otra máquina, que representaba el nodo. Se descargó *Rocks* de la página oficial[2] y se instaló en las máquinas virtuales de *Virtual Box*. El proceso de instalación detallado se puede consultar en el Anexo 2.

Para las pruebas con *Rocks* usando máquinas físicas, se diseñó un pequeño *cluster* basado en tres computadoras Laptop que seguía la siguiente arquitectura.

Las especificaciones de cada una las laptops mencionadas en la Gráfica 1 eran las siguientes: Laptop 1 con el *Frontend* del sistema, corre como server host de *BOINC* y *server HOST* de *ROCKS*. Siendo una HP DV6 con 6 GB de RAM y procesador Core i5; la laptop 2 como *Backend* del Sistema, corre como *slave* de *ROCKS*, siendo una HP DV6 con 6 GB RAM y procesador Core i5; la laptop 3 como *Frontend* del Sistema conectada con *BOINC* a la Laptop 1, siendo esta una HP Dm6 con 4 GB RAM y procesador Core i3.

Usando estas laptops, se llevaron a cabo tres intentos de instalación. En el primero se siguió el tutorial de la documentación oficial, disponible[3]. Durante este intento se encontraron problemas como la falla de detección de los adaptadores de red de las computadoras como *Network cards* y no permitía iniciar la conexión a la Red. También el *frontend* se trabó al intentar realizar la configuración. En el segundo intento, se cambiaron las laptop 1 y 2 de posición en la cadena y se instaló la distribución de *RedHat Linux* que viene con *ROCKS* para ambas. Se siguió el tutorial disponible[4]. Con este nuevo intento se dieron errores en la lectura de discos de instalación y una instalación corrupta en la laptop 2. Durante el tercer intento, la instalación de *frontend* en la laptop 1 fue exitosa, mientras que la búsqueda del *backend* de la laptop 2 fue fallida, dado que no se detectaba ninguna aplicación en la lista. Por lo cual se decidió continuar con la implementación del *cluster* de *BOINC* y se descartó *ROCKS*.

Para las pruebas realizadas con *BOINC*, la instalación y configuración inicial se realizó en una computadora modelo Dell Optiplex 780 con procesador Core 2 Duo E75002.93GHz, 1.7 GB de RAM con sistema operativo CentOS 7. Dentro de ésta se actualizaron los paquetes necesarios antes de proceder a la instalación de *BOINC*. Se procedió a configurar *BOINC* usando *Caching*, para construir la aplicación dentro de la computadora. Se realizó la configuración de *Remote Login*, la

[2] http://www.rocksclusters.org
[3] http://www.rocksclusters.org/presentations/tutorial/tutorial-1.pdf.
[4] http://www.rocksclusters.org/rocks-documentation/4.1/install-frontend.html.

configuración de acceso restringido únicamente a *BOINCTasks PC IP Address*, y se agregó el puerto al *Firewall* para la comunicación de *BOINCTasks* entre clientes *BOINC*. Esta computadora tuvo el papel de *esclava*, mientras que el *master* fue instalado en un servidor Dell PowerEdge R720 con procesador Intel Xeon de 2 GHz y 32GB de RAM, con sistema operativo Ubuntu 16.04.01. Las configuraciones que se llevaron a cabo pueden ser encontradas fácilmente en la página oficial de BOINC. Tras la finalización de la instalación de las herramientas, se verificó si realmente existía conexión entre el servidor y la máquina cliente, resultando ser exitosa dicha conexión.

## Fase 2: Implementación de *BOINC* dentro de una red local

Se instaló el sistema operativo *Ubuntu* en las máquinas a usar como esclavas. Estas máquinas tenían procesador Intel Pentium D de 3.00Ghz y memoria RAM de 2.5GB hasta 3.00GB. Cada una de las computadoras se nombró como "Slave_N" donde N era el número de computadora esclava utilizada. Se tuvieron 3 computadoras disponibles para estas pruebas. En estas se instaló *BOINC client*.

Se revisó la instalación de *BOINC* en el servidor ya mencionado, dado que este tenía la finalidad de ser *master*. Con el propósito de realizar de manera más eficiente las instalaciones, éstas se realizaron dentro de un ambiente de contenedores *Docker* con el cual las dependencias eran instaladas dentro de la máquina. Para realizar la comunicación con el servidor se utilizó *SSH* (terminal segura). Para lograr que el *cluster* funcionara correctamente se realizaron algunos pasos adicionales a la configuración sugerida en las páginas oficiales de instalación para acoplar la instalación al ambiente de contenedores y para corregir algunos errores que se encontraron en el proceso. La bitácora de estos cambios a detalle puede consultarse en el Anexo 3.

Luego se procedió a realizar varias pruebas con el sistema en línea. Las pruebas fueron sencillas dado que solamente se deseaba comprobar que el *Master* distribuye las tareas en cada una de las máquinas *Slave* que se encuentran dentro de la red. La prueba constó en convertir una cadena de caracteres de minúsculas a mayúsculas.

## Fase 3: Implementación de *BOINC* en *AWS*

Se levantó una instancia en *AWS*, con procesador Xeon 2.3Ghz, 1GB de RAM, y 30GB de disco duro. Se instaló *Docker* y *BOINC*. Con esto se usó un *Docker* para *Apache*, para *BOINC* y un *Docker* para *MySQL*. En este servidor se tuvo abierto el puerto 80 para que pudiera ser accedido en forma remota desde la internet. Las tres imágenes de *Docker* se conectaron para facilitar el manejo y conexión de las mismas. El uso de *Docker* facilitó el levantado del servidor de forma remota, dado que al ejecutar la instrucción *docker-compose build* se crea una nueva instancia del servidor y se puede ponerlo a correr. Al ejecutar la instrucción *docker-compose up-d* se levantan las imágenes de *Docker* mencionadas en detachment. Dentro del servidor se crean pequeñas imágenes de sistemas operativos, lo más minimalistas posible, para que puedan ser distribuidas a los clientes.

Del lado de los clientes, es necesario instalar el *BOINC client manager* juntamente con una sesión de *VirtualBox* para poder manejar las diferentes peticiones de parte del *master*. Se puede descargar *BOINC con VirtualBox*[5]. Es necesario configurar el cliente para que se conecte al servidor levantado de Amazon. Para ello es necesario crear una cuenta en la dirección siguiente[6]. Una vez hecho esto, se necesita agregar el proyecto al *Boinc Manager*[7] para emplearlo como URL del proyecto.

Con el ambiente de *BOINC* levantando en un servidor público, se dio la necesidad de especificar los pasos necesarios, de forma más clara y precisa, si una persona quisiera colaborar con el *cluster* implementado o bien si alguien deseara solicitar que algún algoritmo fuera implementado en el sistema. Los pasos a seguir se muestran en la Gráfica 2.

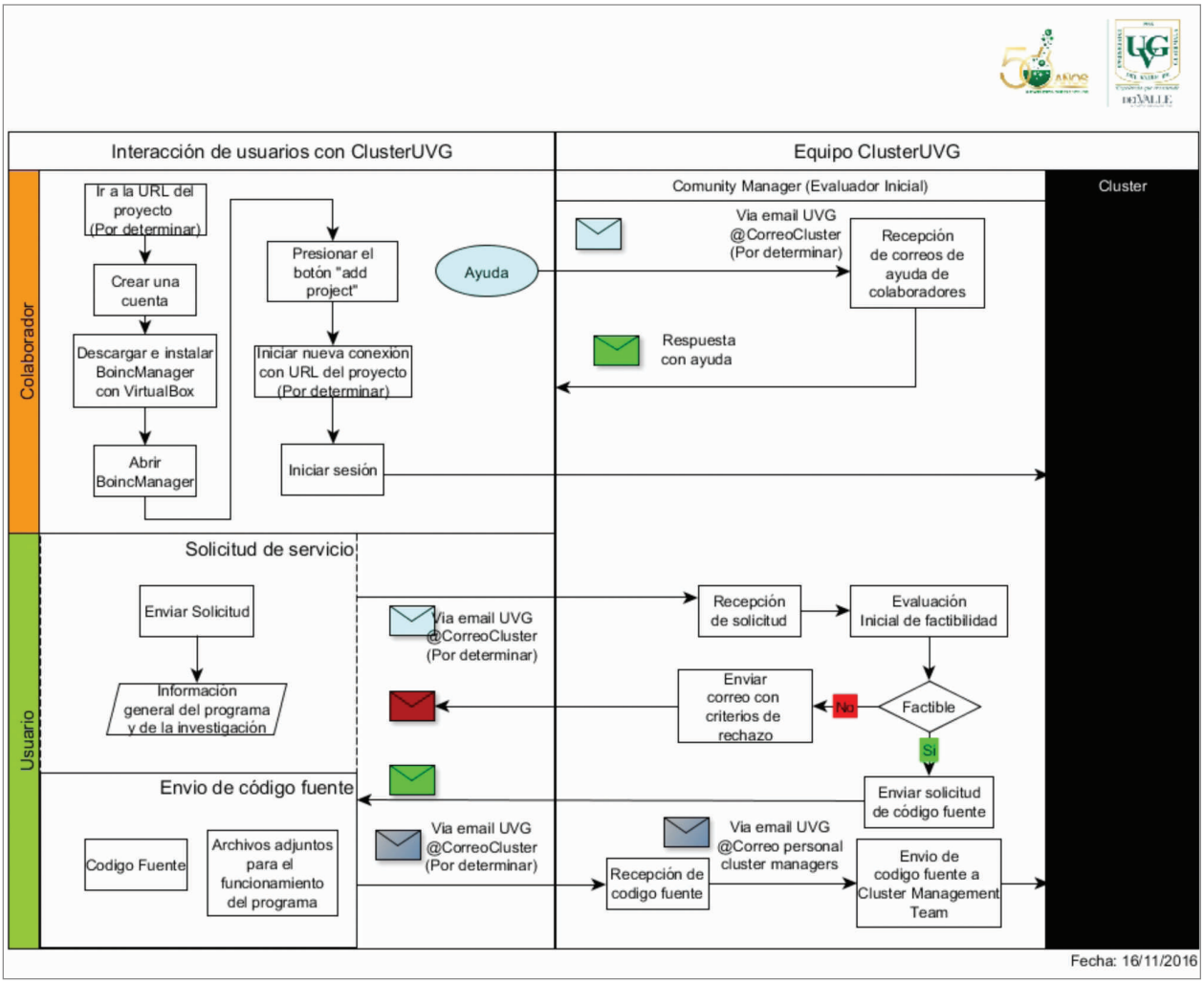
La Gráfica 3 aclara la forma en que un administrador, que tiene conocimiento de *Docker*, puede levantar un nuevo grupo de tareas para un algoritmo requerido.

## Fase 4: Pruebas Usando *BOINC*

Con la finalidad de comprobar la funcionalidad del *cluster* implementado, se tomó un problema relativamente sencillo de las matrices de Hadamard. El problema consiste en calcular las matrices de orden M usando el algoritmo de Goethals-Seidel, que utiliza secuencias de Turyn (Horadam, 2007:263). Se dice que las matrices n x n son matrices de Hadamard, llamadas así en honor a Jacques Hadamard, si cumplen con las siguientes propiedades: Cada elemento es 1 o -1 tal que sus filas sean ortogonales, cada vector fila de la matriz tiene longitud n y al realizar el producto escalar de dos filas distintas cualesquiera el resultado es 0. A simple vista una matriz de Hadamard puede parecer sencilla, pero existen muchos problemas sin resolver, uno de los más comunes es el de encontrar todos los valores de n para los que existe una matriz de Hadamard n x n (Apostol, 1996:86). Las matrices de Hadamard tienen muchas aplicaciones, entre ellas el campo de las telecomunicaciones y procesamiento de señales, estadística, espectrografía, criptografía, modulación de señales, entre otras (Horadam, 2007:92)

El algoritmo que busca resolver el problema mencionado fue desarrollado en un *script* utilizando el lenguaje *Python* en su versión 3. Se realizaron pruebas utilizando una sola computadora de procesador Core i7, con 12GB de RAM y sistema operativo *Arch Linux,* y luego se resolvió el mismo problema a través de

[5] http://boinc.berkeley.edu/download.php.
[6] http://www.cluster.uvg.edu.gt/boincserver/.
[7] http://www.cluster.uvg.edu.gt/boincserver/.



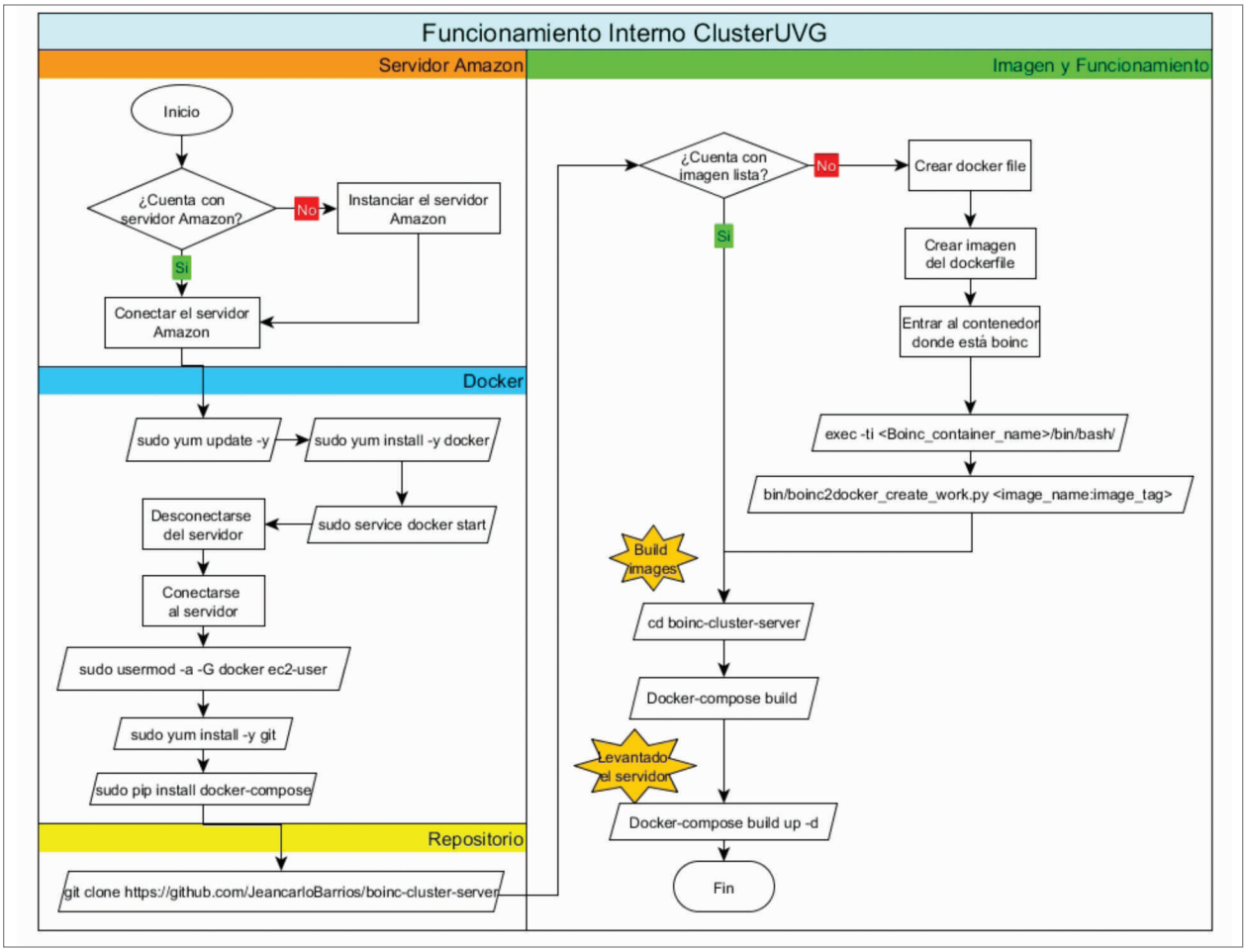

**Gráfica 2.** Diagramas de Interacción de usuarios y solicitud de servicios.

siete computadoras conectadas a la red del *cluster*, cada una con diferentes especificaciones. El tiempo transcurrido fue tomado como el principal factor de comparación en la resolución del problema planteado. En la computadora individual se corrió el script en un ciclo de 8 iteraciones, mientras que en las pruebas usando el cluster se subdividió el trabajo en 8 computadoras distintas conectadas a la red.

## Resultados y discusión

Para las pruebas de *Rocks*, el hecho que las máquinas fueran virtualizadas hizo que la conexión entre los diferentes nodos fuera más sencilla. Esto puede ser explicado por el hecho que se encuentran en ambientes controlados por la máquina *host*, sin externalidades y ruido que se pueda dar en la red externa. Por otro lado, con las máquinas físicas los inconvenientes que surgieron fueron probablemente debidos a ruido dentro de la red, o bien a una configuración errónea de los puertos a ser utilizados.

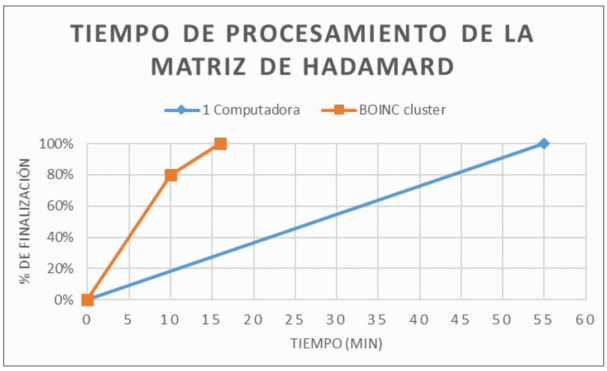
El uso de *BOINC* para la implementación de un *cluster* de computadoras en la Universidad del Valle de Guatemala, resultó ser efectivo dado que se logró distribuir tareas en las diferentes computadoras que se conectaban a la red. De igual modo, el uso de *Docker* en las implementaciones de *BOINC* para al servidor, hizo que fueran más fáciles de colocar en línea. Esta herramienta permitió que fuera más sencillo mover la instancia del *master* de un servidor de red local a uno público para acceso remoto desde internet. Utilizar la versión de *BOINC*, con *VirtualBox*, para los *esclavos* y su conexión al *master* amplió la cantidad de tareas que el *cluster* puede resolver. Incluir virtualización permite que cualquier computadora pueda conectarse al *cluster* y procesar tareas sin importar su sistema operativo. Además, debido a que la versión de *BOINC* utilizada incluye *VirtualBox*, el tiempo necesario para que un colaborador pueda participar en la resolución de tareas es menor. Otra característica llamativa de *BOINC* es la capacidad de reconocer cuando las computadoras de los colaboradores no están siendo utilizadas, y aumentar la cantidad de recursos a los que el *cluster* puede accesar.




**Gráfica 3.** Pasos a seguir para implementar nuevas instancias del *master*.

Para las pruebas realizadas, primero se probó ejecutar el mismo script para las matrices de Hadamard en una sola computadora, verificando que tardó 55 minutos. Luego se crearon 8 unidades de trabajo de *BOINC* y se colocaron en el servidor para que pudieran ser distribuidos a las computadoras que estaban colaborando con el *cluster*. En esta modalidad, el proceso tomó 18 minutos, una reducción del 67% del tiempo utilizado con una sola computadora. Esta reducción de tiempo concuerda con lo esperado, pues se tenía a diferentes computadoras trabajando paralelamente en diferentes tareas que eran distribuidas por el *master*. La ejecución fue más corta e incluso más eficiente para la máquina encargada de la tarea principal, el *master* como se puede observar en la Gráfica 4. Durante las pruebas se observó que una computadora *esclava* no aceptaba tareas al mismo ritmo que las otras. Esto puede deberse a fallos en la configuración de la distribución de tareas de *BOINC* o a la carga de procesamiento de otros programas en la computadora *esclava*.

Se sugiere que la implementación de otro tipo de algoritmos sea llevada a cabo con el objetivo de poder ser probados dentro de este *cluster* y así reafirmar su utilidad para la comunidad universitaria. Algunas recomendaciones adicionales son:




**Gráfica 4.** Tiempo de procesamiento de la Matriz de Hadamard de prueba.

- Buscar alternativas para poder hacer las imágenes, dadas por el servidor, lo más livianas posibles para que el tiempo de descarga de tareas y requisitos de cada uno de los clientes sea lo más rápido posible, y que puedan proceder a ayudar en la carga computacional a la mayor brevedad.

- Implementar este tipo de cluster con los laboratorios de computación del campus dado que estos, al estar cierto tiempo sin uso, pueden colaborar con el *cluster* significativamente, en términos computacionales.
- Utilizar el protocolo *https* para mejorar la seguridad del *cluster*.
- Realizar pruebas de escalabilidad con más computadoras para determinar las limitaciones de la implementación realizada.

## Conclusiones

El uso de *Docker* facilitó y agilizó el proceso para designar a una computadora como *master*. De no utilizar *Docker*, el proceso hubiera sido tedioso y difícil de modificar cuando se deseaba hacer cambios. Además, el uso de *BOINC* para la implementación de un *cluster* facilitó dos aspectos clave: la inserción de nuevos colaboradores a la red y la distribución automática de problemas computacionales (algoritmos) desde el *master* hacia los *slaves*. Con el *cluster* ya implementado en *AWS*, se pudo probar la efectividad del *cluster* al ejecutar un problema de matrices de Hadamard. Este trabajo sienta las bases para una implementación futura de un *cluster* a mayor escala que pueda ofrecer beneficios de procesamiento computacional a investigadores y estudiantes de la Universidad del Valle de Guatemala.

## Agradecimiento

Se agradece a la Asociación de Estudiantes de Ingeniería en Ciencias de la Computación y Tecnologías de la Información 2016 de la Universidad del Valle de Guatemala, por haber facilitado parte del equipo en que se llevó a cabo las primeras pruebas. De igual modo, se extiende el agradecimiento al licenciado Mario Gómez, por facilitarnos el problema que se usó para realizar las pruebas del *cluster*.

## Bibliografía

BOINC (2008) *Create a Virtual Campus Supercomputing Center*[8].

Cesarini, F., Vinoski, S. (2016) *Designing for Scalability with Erlang/OTP: Implement Robust, Fault-Tolerant System* 1a ed. Estados Unidos. O'Reilly Media, Inc. 482 págs.

Donahue, G. (2007) *Network Warrior* O'Reilly. pag. 5.

Hwang, K., Fox, G., Dongarra, J. (2012) *Distributed and Cloud Computing: From Parallel Processing to the Internet of Things* 1a. ed. Estados Unidos. Elsevier, Inc. 672 págs.

Milone, D.H., Azar, A.A., Rufiner, L.H. (2002) *Supercomputadoras basadas en "Clusters" de PCs* Revista Ciencia, Docencia y Tecnología, 13 (25), 173-208.

Tao, F., Zhang, L., Laili, Y. (2015) *Configurable Intelligent Optimization Algorithm: Design and Practice in Manufacturing* 1a ed. Cham, Suiza. Springer International Publishing. 361 págs.

Horadam, K.J. (2007) *Hadamard Matrices and Their Applications* 1a. ed. Estados Unidos. Princeton University Press. 263 págs.

Apostol, T. (2006) *Cálculo: Funciones de varias variables* 2da. ed. Barcelona, España. Editorial Reverté S.A. 800 págs.

Sloan, J. (2005) *High Performance Linux Clusters with OSCAR, Rocks, OpenMosix, and MPI* 1a. ed. Estados Unidos. O'Reilly Media, Inc. 350 págs.

## ANEXOS

### Anexo 1: Glosario de términos

*Backend:* Parte de un sistema que realiza las operaciones lógicas y se comunica con la base de datos y el frontend.

*BOINC:* Herramienta para la distribución de tareas de forma comunitaria a distintas computadoras

*Cluster:* Conjunto de computadoras que donan poder de procesamiento para realizar tareas específicas de forma paralela.

*Docker:* Herramienta para la creación de contenedores independientes autosostenibles.

*Frontend:* Parte de un sistema que interactúa con un usuario.

*Slave:* Computadora encargada de realizar tareas distribuidas por la computadora maestra. También referido como esclava en esta publicación.

*Master:* Computadora encargada de distribuir las tareas entre las computadoras del cluster.

### Anexo 2: Detalles de Instalación de *ROCKS* en máquinas virtuales

Para instalar *Rocks*, se descarga el instalador en formato ISO del sitio oficial de *ROCKS*, este instalador se basa en una distribución de *Linux* llamada *Red Hat*. En el caso del nodo, no se utilizó el archivo ISO para instalar *Rocks*, sino que se conectó al *frontend* por medio de una red interna. Esta red interna es virtual, y representa una conexión por medio de cable *crossover* entre los dos puertos Ethernet de las computadoras. Para este proceso, es necesario ingresar a la terminal del *frontend* y escribir "insert-ethers", luego se debe elegir la opción "compute". Cuando el *master* se encuentre buscando el nodo se debe encender la máquina virtual del nodo. Cuando esta se encienda, el *frontend* la detectará y le enviará los archivos necesarios para instalarse como un nodo. Al momento que el nodo se encuentre conectado, se debe presionar F8 para salir de la búsqueda de nodos y esperar a que *Rocks* se instale en el nodo. Luego de haberse instalado, se puede verificar la conexión de las máquinas[9]. Cabe mencionar que en algunos momentos la cantidad de RAM aumenta a 2 GB. Este momento representa el instante cuando el nodo se conectó al *frontend*.

[8] https://boinc.berkeley.edu/trac/wiki/VirtualCampusSupercomputerCenter#no1 [5/14/16].
[9] http://localhost/ganglia/.

**Anexo 3: Detalles de instalación de *BOINC* en una red local.**

Para que *BOINC* corriera de forma correcta, se necesitó cambiar el *Hostname* con el nombre de la computadora en el archivo config.xml en la ruta */bar/lib/boinc-server/boincproject/config.xml*.

La instrucción para iniciar *BOINC* es*./bar/lib/boinc-server/boincproject/bin/start*. Además, para iniciar la base de datos se necesita ejecutar la instrucción *service mysql start*. También se configuró la contraseña de Apache dentro del servidor. Nótese que cuando se inicia el servidor Apache puede llegar a ser necesaria la instrucción*./bin/strat* para iniciar tanto el *feeder*, *transitioner* y *file deleter*.

Cuando se crea una nueva aplicación para procesar, se debe informar a la base de datos acerca de la nueva aplicación para que esta se actualice. Esto se hace modificando el archivo version.xml con las instrucciones encontradas en la siguiente dirección[10]. Siguiendo la guía, se encontró que era necesario eliminar las líneas *"cat version.xml"* y *"fi"* del archivo *version.xml,* para permitir el funcionamiento de la instrucción *./bin/update_versions*.

Para la creación de la nueva unidad de trabajo, se utilizó la instrucción *strace*. Sin embargo, para que ésta funcionara fue necesario agregarla como parámetro al ejecutar la instrucción *run* de *Docker*. A dicha instrucción se le agrega *--security-opt seccomp:unconfined*. A pesar que esto hace más inseguro a *Docker,* ayuda a que *BOINC* pueda correr adecuadamente.

[10] https://wiki.debian.org/BOINC/ServerGuide/AppDeployment.