BOINC-MGE
Mobile Grid Extension

Arturo Segundo García Payares

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
MAESTRIA EN INGENIERIA DE SISTEMAS Y COMPUTACIÓN
BOGOTÁ, D.C.
2019

BOINC-MGE
Mobile Grid Extension

Autor:

Arturo Segundo García Payares

MEMORIA DEL TRABAJO DE GRADO REALIZADO PARA CUMPLIR UNO
DE LOS REQUISITOS PARA OPTAR AL TITULO DE
MAGÍSTER EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

Directora:

Mariela Josefina Curiel Huérfano

Comité de evaluación del trabajo de grado:

Leonardo Flórez Valencia

Harold Enrique Castro Barrera

Página web del Trabajo de Grado

https://livejaverianaedu.sharepoint.com/sites/Ingsis/TGMISC/193001

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
MAESTRIA EN INGENIERIA DE SISTEMAS Y COMPUTACIÓN
BOGOTÁ, D.C
Noviembre, 2019

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
MAESTRÍA EN INGENIERIA DE SISTEMAS Y COMPUTACIÓN

Rector Magnifico

Jorge Humberto Peláez, S.J.

Decano Facultad de Ingeniería

Ingeniero Lope Hugo Barrero Solano

Director Maestría en Ingeniería de Sistemas y Computación

Ingeniera Angela Carrillo Ramos

Director Departamento de Ingeniería de Sistemas

Ingeniero Efraín Ortíz Pabón

Artículo 23 de la Resolución No. 1 de Junio de 1946

"La Universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus proyectos de grado. Sólo velará porque no se publique nada contrario al dogma y la moral católica y porque no contengan ataques o polémicas puramente personales. Antes bien, que se vean en ellos el anhelo de buscar la verdad y la Justicia"

Agradecimientos

A mi esposa e hijo, por su paciencia y apoyo incondicional.

A mi tutora, la profesora Mariela, por su tiempo y valiosos aportes que me guiaron para finalizar con éxito mi trabajo de grado.

Contenido

Abstract

Adding mobile devices to a grid brings some challenges, like device disconnections and battery management that need to be addressed. BOINC as a grid platform that enable users to use their mobile phones as computational devices, does not resolves those issues. BOINC-MGE, presented in this document, is a BOINC extension that solves these challenges by adding two job scheduling strategies, one that use predictive models and the other that uses a replication algorithm.

Resumen

La incorporación de dispositivos móviles en una grid trae consigo desafíos que deben resolverse, tales como las desconexiones de los usuarios y la gestión eficiente de la batería; BOINC, una de las plataformas para el despliegue de proyectos de grid permite la incorporación de dispositivos móviles como plataformas de cómputo, pero no resuelve los desafíos mencionados. En el presente trabajo se presenta y desarrolla BOINC-MGE, una extensión que da solución a dichos desafíos mediante la incorporación de dos estrategias de planificación de trabajos, una basada en modelos predictivos y otra basada en replicas.

Resumen ejecutivo

La computación en grid como herramienta para la división y resolución de problemas ha sido usada desde hace muchos años, a su vez, el aumento en las capacidades de los dispositivos móviles ha permitido su uso en la grid no solo como terminales para acceso a recursos sino también como plataformas de ejecución de trabajos, sin embargo, para que tanto la grid como los dispositivos móviles logren obtener el mayor provecho a esta integración, diferentes tipos de desafíos deben ser resueltos. En particular dos de tipo técnico son tenidos en cuenta en el presente trabajo de investigación: la administración eficiente de los recursos de batería de los dispositivos y garantizar la culminación de los trabajos en escenarios donde los usuarios regularmente se desconectan.

Varios autores han identificado que la mejor forma para la solución de dichos desafíos es la aplicación de estrategias de planificación de trabajos que consideren las características particulares de los dispositivos móviles y sus patrones de uso. Estas estrategias pueden ser principalmente de tipo preventivas o reactivas. Las primeras son aplicadas antes de enviar los trabajos a los dispositivos y consisten en la implementación de modelos predictivos para hacer una selección adecuada del dispositivo móvil de acuerdo con su nivel de batería o en la generación de réplicas para anticiparse a la no finalización de tareas en un dispositivo por problemas de desconexiones o carga. Las estrategias reactivas por otro lado actúan una vez que son asignados los trabajos; mediante la aplicación de mecanismos de migración de trabajos entre los dispositivos, buscan garantizar la finalización de las tareas asignadas ante eventos inesperados de desconexión.

BOINC es una plataforma grid de computación voluntaria que permite la integración de dispositivos móviles como herramientas de computo. Esta característica, sumada a su gran popularidad a nivel mundial y a su flexibilidad para poder integrar una gran variedad de aplicaciones, entre otras razones, fueron suficientes para ser seleccionada en trabajos de grado anteriores como la mejor opción para la implementación de una grid móvil para el procesamiento de imágenes médicas. Sin embargo, a pesar de permitir dispositivos móviles, el algoritmo de planificación de BOINC no resuelve en su totalidad los desafíos planteados. El presente trabajo surge como una solución a dichos desafíos dentro de BOINC, mediante el diseño e implementación de BOINC-MGE, una extensión que mejora las funcionalidades de BOINC para hacerla más compatible con este tipo de grids, a través de la incorporación de dos estrategias de planificación del tipo preventivas. Una de las estrategias de planificación de BOINC-MGE usa un modelo predictivo que calcula la tasa de consumo promedio de batería de los dispositivos y realiza la planificación de los trabajos usando dicha información. La otra estrategia incorporada está basada en *Aprendizaje Reforzado*; esta estrategia, llamada RL REPL busca garantizar la culminación de los trabajos en escenarios de desconexión a la vez que se reduce el consumo promedio de batería en los dispositivos mediante un esquema de replicación de trabajos. Las dos estrategias fueron seleccionadas luego de un proceso de evaluación de varias alternativas propuestas por diferentes autores. La evaluación fue realizada haciendo uso de una metodología para la toma de decisiones de CMMI.

Adicionalmente a las estrategias de planificación se desarrolló una API para facilitar a futuro la integración de nuevas estrategias o la modificación de las dos presentadas en BOINC-MGE.

La validación y evaluación de la extensión BOINC-MGE se realizó mediante la ejecución de un diseño experimental divido en dos partes: una primera parte en la que se hizo la evaluación de la estrategia basada en modelos predictivos y otra para la evaluación de la estrategia basada en réplicas. En el primer diseño experimental se compararon variables como el consumo de energía y el tiempo total de procesamiento de trabajos, usando las estrategias de planificación de BOINC-MGE y BOINC en contextos de poca y mucha batería disponible en los dispositivos móviles. Los resultados obtenidos mostraron que BOINC-MGE mejora los tiempos de ejecución y el consumo promedio de batería de los dispositivos cuando se actúa en un escenario donde la batería es limitada. El segundo experimento evaluó la estrategia de planificación basada en réplicas de BOINC-MGE comparada con distintas configuraciones de replicación de BOINC en 4 escenarios de ejecución diferentes: uno ideal sin desconexiones en los dispositivos y otros tres con desconexiones a diferentes frecuencias generadas de forma aleatoria. En todos los escenarios, BOINC-MGE obtuvo mejores resultados de tiempo de ejecución y de consumo de batería comparado con las estrategias de BOINC. También se realizó un conjunto de pruebas para verificar el impacto de las nuevas estrategias en el uso de recursos del servidor de BOINC. Los resultados mostraron que las nuevas estrategias aumentan el uso de algunos recursos en el servidor, pero de una forma poco significativa y que puede tolerarse con las características actuales de hardware.

Finalmente, se identifican nuevas oportunidades de trabajos que pueden ser realizados a futuro para aumentar las características de BOINC en el contexto de las grids móviles, así como también la ejecución de experimentos sobre BOINC-MGE en otro tipo de escenarios.

1 Introducción

El uso de los dispositivos móviles se ha incrementado de forma considerable alrededor del mundo en los últimos años. De acuerdo con estudios de mercado realizados por Ericsson Mobile[1], existían en el año 2016 alrededor de 3900 millones de usuarios de teléfonos inteligentes y se estima que para el año 2022 ésta cantidad aumente hasta los 6800 millones en todo el planeta. Este crecimiento y el hecho de que dichos dispositivos tengan características de procesamiento y recursos cada vez más altos, los convierte en candidatos ideales para ser usados como plataformas de computo de forma similar en la que se usan computadoras personales en proyectos como BOINC[2] o HTC Condor[3].

Aunque no se trata de una idea nueva, las llamadas grids móviles han venido siendo objeto de estudio con más frecuencia en los últimos años [4][5][6], esto se debe, por un lado al mejoramiento de las características ya mencionadas, y al crecimiento en la cantidad y variedad de dispositivos conectados a internet; fenómeno conocido como *internet de las cosas* o IoT por sus siglas en inglés. Además, nuevos conceptos y tecnologías tales como *edge* o *fog computing* también han venido ganando popularidad, los cuales pretenden aprovechar estos dispositivos más cercanos a los usuarios para realizar tareas de procesamiento[7] y disminuir la carga y tiempos de latencia de red en los servidores ubicados en la nube.

Diferentes casos de uso que muestran los beneficios de incorporar dispositivos móviles como plataformas de computo han sido presentados por otros autores [8][9][10][11]; en la misma línea, en la Universidad Javeriana se realizaron dos trabajos de pregrado: en el año 2015 en el proyecto de grado *Grids accesibles*[12] se exploró la posibilidad de implementar grids mediante dispositivos móviles, seleccionando la plataforma BOINC como la más adecuada debido a su madurez y amplio uso a nivel global. En el año 2016, el trabajo de grado *Grid móvil para procesar imágenes médicas* [13][14] demostró que es posible el uso de dispositivos móviles para el procesamiento de imágenes, obteniendo tiempos de computo muy similares a los logrados con computadores de escritorio.

Sin embargo, en [14] también se pudieron evidenciar algunos de los desafíos advertidos por otros autores [15][16][17], los cuales deben ser encarados al momento de integrar dispositivos móviles a la grid y que no son resueltos particularmente en la plataforma BOINC; dichos desafíos tienen que ver principalmente con la probabilidad de que los trabajos no sean finalizados en el tiempo adecuado cuando son enviados para su ejecución debido a circunstancias, tales como: a) agotamiento de la batería, b) desconexiones de los dispositivos por decisión de los usuarios o de forma automática debido a su propiedad inherente de movilidad y c) recursos insuficientes (RAM, Disco, CPU).

BOINC es una grid para la computación voluntaria y aunque permite los dispositivos móviles como plataformas de ejecución, aún carece de ciertas características propias de una grid móvil, tales como un algoritmo de planificación que considere el estado de la batería del dispositivo para la asignación de recursos. En este trabajo se propone dar respuesta a este reto, incorporando a BOINC algoritmos de selección que consideren el estado actual de la batería en los dispositivos móviles y posibles estados de desconexión, ya que en la actualidad se considera principalmente el uso de CPU.

El presente documento describe las tareas llevadas a cabo para alcanzar el objetivo planteado; en el capítulo **2. Descripción general** se presenta, de forma más detallada el objetivo del trabajo de grado, su impacto y utilidad; en el capítulo **3. Marco metodológico** se explican las metodologías usadas para el desarrollo del proyecto y cada una de las fases que lo componen; el capítulo **4. Marco teórico** presenta conceptos fundamentales para entender el problema planteado y la solución propuesta; en el capítulo **5. Análisis y selección de estrategias de planificación** se detalla el proceso de evaluación y selección de las estrategias de planificación a incorporar a BOINC, así como una descripción de todas las opciones analizadas; en el capítulo **6. Arquitectura y componentes BOINC-MGE** se describe la arquitectura general de BOINC-MGE y los componentes que fueron necesarios introducir y/o modificar en BOINC para generar la extensión propuesta; el capítulo **7. API para integración de algoritmos de planificación en BOINC-MGE** describe una interfaz de programación de aplicaciones que se agregó como parte de la extensión desarrollada, la cual permite la incorporación de nuevas estrategias de planificación de forma más sencilla; en el capítulo **8. Estrategias de planificación en BOINC-MGE** se describen los algoritmos que soportan las estrategias de planificación seleccionadas e incorporadas a BOINC-MGE haciendo uso de la API de desarrollo; el capítulo **9. Pruebas y evaluación de resultados** describe el proyecto de prueba de BOINC creado para la validación de las estrategias y el protocolo experimental diseñado para la verificación de la extensión MGE. Además, se discuten los resultados obtenidos en comparación con el modelo actual de BOINC en un escenario de grid móvil con teléfonos inteligentes de diferentes características; finalmente, en el capítulo **10. Conclusiones** se resumen los hallazgos obtenidos durante el trabajo de investigación, así como recomendaciones y líneas de acción a futuro.

2 Descripción general

La computación en grid[18] es un modelo de computación distribuida en el cual se hace uso de un conjunto de computadores no acoplados, dispositivos de almacenamiento e instrumentos; los cuales se encuentran interconectados y gestionados de forma transparente por un componente de middleware para la resolución de problemas científicos o comerciales. La computación voluntaria[19] es un tipo de computación en grid en el cual un grupo de personas (también llamados voluntarios) aportan sus propios recursos a proyectos de computación distribuida para que sean usados para almacenamiento o tareas de computo.

Las grids móviles son otro modelo de computación distribuida que extiende el concepto de computación en grid a dispositivos móviles, tales como los teléfonos celulares. Inicialmente, la integración de dispositivos móviles a la grid estuvo reducida a usarlos como herramientas de acceso a las interfaces de administración de una grid convencional, dado que las capacidades con las que contaban dichos dispositivos no los hacían viables para ser integrados como herramientas de cómputo[20]. En la actualidad las capacidades de los dispositivos móviles han aumentado y es cada vez más común integrarlos para aprovechar sus capacidades de almacenamiento y computo en sistemas de computación en grid y en nuevas tecnologías y paradigmas tales como el internet de las cosas, computación en la niebla (*fog computing*) y/o computación en el borde (*edge computing*)[7][21][15]. Sin embargo, al usar estos dispositivos como plataformas de computo, surgen diferentes inconvenientes, tanto técnicos como humanos que deben resolverse. Dentro de los desafíos humanos se encuentran, por ejemplo, el convencimiento a los usuarios para que aporten sus dispositivos, promover la permanencia en la grid durante el tiempo que sea necesario y evitar comportamientos maliciosos. En este trabajo abordamos los desafíos técnicos, los cuales se describen a continuación.

2.1 Planteamiento del problema

Dentro de los inconvenientes técnicos que han sido identificados con anterioridad por varios autores con respecto a la integración de dispositivos móviles como elementos de procesamiento en una grid [22][23][12], se destacan principalmente:

- **Consumo de energía**: Cada operación llevada a cabo por la CPU o cada byte transmitido, requiere un consumo en la batería de los dispositivos, la cual sigue siendo limitada.
- **Disponibilidad**: Al estar conectados mediante redes inalámbricas y debido a la naturaleza móvil de los dispositivos, es muy probable que el tiempo disponible en la grid no sea constante.

Estos inconvenientes pudieran tener un efecto adverso principalmente en el tiempo de ejecución de los trabajos enviados, siendo el peor de los casos su no culminación. Si se acaba la batería del dispositivo o si el usuario se desconecta (voluntaria o involuntariamente), las tareas no culminan. Es importante, por lo tanto, que las grids móviles tengan en cuenta estas limitaciones e inconvenientes y actúen de tal forma que puedan garantizar la confiabilidad en cuanto a los tiempos de finalización de trabajos.

Existen múltiples plataformas que permiten la implementación de grids, sin embargo, tal como se evidencia en uno de los trabajos de grado previos[12] las opciones son más limitadas cuando se desea realizar la integración de dispositivos móviles. Entre las opciones disponibles,

BOINC fue seleccionada como la plataforma para la implementación de una grid móvil para el procesamiento de imágenes médicas[14]. BOINC fue escogida por cumplir con los siguientes requisitos: posibilidad de ejecutar tareas escritas en C++, ejecución de tareas en dispositivos móviles, disponibilidad del código fuente de la plataforma para realizar modificaciones y calidad en la documentación técnica de la plataforma.

Sin embargo, BOINC carece de ciertas características propias de una grid móvil, como un algoritmo de planificación que considere el estado de la batería del dispositivo para la asignación de recursos o la desconexión de los usuarios. Por otro lado, al tratarse originalmente de un proyecto que se rige mediante la filosofía de computación voluntaria[19] no se considera una prioridad que los trabajos concluyan rápidamente, uno de los requerimientos más importantes para BOINC es la confiabilidad de los recursos computacionales voluntarios, es decir que los resultados devueltos no sean erróneos.

El trabajo **BOINC-MGE (Mobile Grid Extension)** descrito en la presente memoria se propone dar respuesta a los retos planteados, incorporando a BOINC algoritmos de selección que tomen en cuenta el estado actual de la batería en los dispositivos móviles y posibles estados de desconexión, ya que en la actualidad se considera principalmente el uso del CPU. Adicionalmente, se agrega una interfaz de desarrollo (API) que permita agregar a futuro otras estrategias de planificación.

2.2 Impacto esperado

El trabajo desarrollado consta de dos componentes, la API de desarrollo para BOINC y la extensión BOINC-MGE para grids móviles; cada uno brinda herramientas importantes que pueden ser aprovechadas en diferentes campos como la investigación y docencia, salud, empresas, educación, etc.

- En el campo de la investigación y docencia se brinda la posibilidad de aprovechar la API de desarrollo generada para agregar de forma más sencilla a BOINC algoritmos de planificación y distribución de tareas en grids móviles.
- BOINC-MGE puede ser usado para el despliegue y configuración de grids móviles que permitan la ejecución de tareas que requieren mucho procesamiento en situaciones donde no se cuente con recursos suficientes. Algunos ejemplos pueden ser hospitales, centros de salud y/o instituciones educativas con poco acceso a la tecnología o escasos recursos económicos; estas instituciones pueden aprovechar los dispositivos móviles que pudieran brindar los usuarios que acceden a sus servicios o sus propios empleados (pacientes, estudiantes, profesores, personal médico, etc.).
- En el ámbito empresarial una solución como BOINC-MGE abre la posibilidad de crear grids móviles privadas que vendan capacidad de cómputo para proyectos privados (Mobile Grid Computing as a Service) haciendo uso de dispositivos móviles alquilados por usuarios que deseen obtener un beneficio económico.

2.3 Objetivos

2.3.1 Objetivo general

Integrar estrategias de planificación de tareas en BOINC que consideren el estado de la batería de los dispositivos móviles y la desconexión de los usuarios en los nodos de ejecución.

2.3.2 Objetivos específicos

- Evaluar estrategias de planificación de tareas en grid móviles que tomen en cuenta el estado de recursos como la batería y estados de desconexión en los dispositivos de ejecución.
- Formular y desarrollar una API sobre BOINC para la implementación de algoritmos de planificación de tareas.
- Implementar dos estrategias de planificación de tareas en BOINC, que tomen en cuenta el estado de la batería y las desconexiones de los usuarios.
- Evaluar, mediante pruebas, las estrategias implementadas usando aplicaciones que hagan uso intensivo del procesador.

3 Marco metodológico

El desarrollo del trabajo de investigación se llevó a cabo a través de diferentes fases, cada una con una metodología distinta y asociada a los objetivos específicos que se pretendían cumplir. La **Figura 1** muestra las fases ejecutadas y las metodologías usadas en cada una de ellas. Las siguientes secciones describen las fases del proyecto.

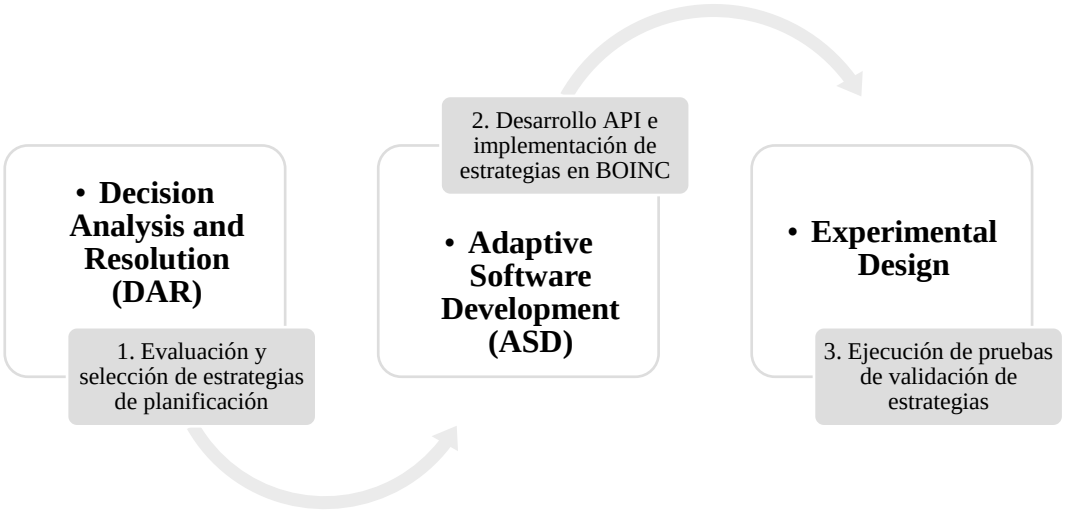


Figura 1. Fases de ejecución y metodologías usadas

3.1 Fase 1: Evaluación y selección de estrategias de planificación

En la primera fase se llevó a cabo la revisión del estado del arte con respecto a grids móviles y sus estrategias de planificación. Luego, a partir de la revisión y análisis bibliográfico se hizo uso de la metodología DAR[24] para la selección de las estrategias que fueron incorporadas en BOINC-MGE.

El propósito de la metodología DAR es analizar posibles decisiones mediante el uso de un proceso formal de evaluación que permite comparar diferentes alternativas basado en criterios bien definidos de evaluación.

Las actividades que se llevaron en esta fase se resumen de la siguiente forma:

- Revisión del estado del arte mediante consultas bibliográficas sobre estrategias de planificación en grids en general y más detalladamente sobre grids móviles.
- Clasificación de las propuestas encontradas teniendo en cuenta las técnicas aplicadas y su aplicabilidad en el contexto de las grids móviles. La forma de clasificación de estrategias de planificación en grids móviles es descrita en el capítulo **4. Marco teórico**.
- Uso de la metodología *Decision, Analysis and Resolution* (DAR) para la selección de las estrategias a implementar en el proyecto BOINC-MGE. Los resultados y detalles de esta actividad son presentados en el capítulo **5. Análisis y selección de estrategias de planificación.**

3.2 Fase 2: Desarrollo del API e implementación de estrategias en BOINC

La fase 2 del trabajo consistió en el diseño e implementación de los dos componentes software del proyecto. Por un lado, la interfaz de desarrollo (API) de estrategias de planificación para grids móviles en BOINC y por otro lado la implementación de las estrategias que serían integradas a BOINC usando dicha API para generar BOINC-MGE. Esta fase fue ejecutada usando el modelo de desarrollo de software por prototipos aplicando los conceptos de la metodología ASD (Adaptive Software Development) [25].

ASD es un modelo de implementación de patrones agiles de funcionamiento cíclico en el cual se aceptan la introducción de cambios y generación de errores a lo largo de las diferentes iteraciones de desarrollo. Consta de tres fases que se repiten de forma cíclica: **especular**: donde se definen el alcance y funcionalidades a desarrollar en el ciclo que se está iniciando; **colaborar:** fase en la que se desarrollan los componentes y funcionalidades comprometidas en la fase anterior; **aprender**: etapa en la que se revisa la calidad del desarrollo hecho y su alineación con los objetivos iniciales. Al finalizar esta etapa se vuelve a la fase **especular** hasta que se han cumplido con todos los objetivos del proyecto.

Las principales actividades de esta fase, llevadas a cabo mediante la aplicación de los conceptos de ASD son:

- Analizar, planear y diseñar la API de desarrollo y la arquitectura sobre la cual se soporta BOINC-MGE.
- Implementar la API de desarrollo en BOINC realizando las modificaciones necesarias en sus componentes internos para adaptarlo a la nueva extensión MGE.
- Haciendo uso de la API de desarrollo implementar las estrategias de planificación mediante la metodología ASD basada en prototipos.
- Realizar pruebas funcionales sobre la API y las estrategias de planificación para verificar su correcta integración con el modelo de BOINC.

Los resultados de esta fase son descritos en los capítulos **6. Arquitectura y componentes BOINC-MGE**, **7. API para integración de algoritmos de planificación en BOINC-MGE** y **8. Estrategias de planificación en BOINC-MGE**.

3.3 Fase 3: Ejecución de pruebas de validación de estrategias

La última fase de la investigación comprendió la definición y ejecución de las pruebas a realizar sobre la extensión BOINC-MGE para verificar el desempeño y utilidad que aportan las estrategias de planificación elegidas en el contexto de las grids móviles.

Se diseñó un proyecto BOINC de prueba y, mediante la metodología del diseño experimental, se generaron y ejecutaron pruebas de tipo factorial con varios factores y niveles. Estas pruebas permitieron medir el impacto de varios factores en diferentes variables respuesta, tales como el tiempo de ejecución de los trabajos, el consumo de batería de los dispositivos, y la cantidad de trabajos ejecutados, entre otras. El capítulo **9. Pruebas y evaluación de resultados** describe el proyecto BOINC generado para realizar los experimentos, las pruebas realizadas sobre las dos estrategias de planificación de BOINC-MGE y el análisis de los resultados obtenidos.

4 Marco teórico

El presente trabajo requirió de la apropiación de conceptos tales como computación en grid, grids móviles, planificación de trabajos, etc., así como la investigación sobre el funcionamiento de la plataforma de computación voluntaria BOINC. En este capítulo se brinda una definición de los conceptos fundamentales que sirven para ayudar al lector a tener una mejor comprensión de las secciones siguientes. De forma similar, se realiza una descripción general del funcionamiento actual de BOINC.

4.1 Conceptos fundamentales

4.1.1 Computación en grid

Iniciativa que involucra la integración de computadoras geográficamente distantes conectadas en red para formar un sistema distribuido de gran escala con el objetivo de resolver un problema de manera coordinada. Al distribuir el procesamiento entre múltiples equipos de menor capacidad, los usuarios de una grid cuentan con las ventajas de obtener mayor capacidad de procesamiento, almacenamiento y ancho de banda que en otro caso solo estarían disponibles a través de las tradicionales supercomputadoras con multiprocesadores, las cuales tienen un costo muy elevado[23] [26][27].

4.1.2 Planificación de trabajos

La planificación de trabajos en grids se refiere al mecanismo mediante el cual se asignan y distribuyen las tareas disponibles en la grid hacia los nodos de ejecución[22]. Se trata de un proceso crítico, ya que una mala estrategia de planificación implicaría el desperdicio o mal aprovechamiento de los recursos disponibles.

4.1.3 Plataforma grid

Herramienta software que permite la configuración y despliegue de una grid. Incluye la implementación de un protocolo de comunicación entre los nodos de procesamiento y el administrador de trabajos de la grid, también incluye al menos un mecanismo de planificación de los trabajos disponibles y la posterior recolección y validación de los resultados retornados por los nodos de procesamiento. El ejemplo más tradicional de plataforma grid es GLOBUS[28].

4.1.4 Administrador grid

Persona o conjunto de personas que se encargan de la administración de los proyectos que se ejecutan dentro de una grid.

4.1.5 Usuario grid

Se refiere a las personas u organizaciones que usan los servicios ofrecidos por la grid. Este tipo de personas no poseen control sobre los trabajos a ejecutar.

4.1.6 Computación voluntaria

Es un tipo de computación en grid en el cual usuarios del público en general donan tiempo ocioso de sus dispositivos (computadores personales o teléfonos inteligentes) para ayudar en la resolución de problemas generalmente científicos y de interés general. Este concepto nació con el proyecto SETI@home[29] en el año 1999, el cual luego evolucionó hacia la plataforma de computación voluntaria genérica BOINC.

4.1.7 Voluntario

Persona u organización que pone a disposición de la grid los recursos (teléfonos inteligentes o computadores personales) que son usados para la ejecución de los trabajos. En el presente trabajo usamos el termino *usuario* para referirnos tanto a los usuarios tradicionales que usan los recursos de la grid como a los voluntarios que ponen a disposición sus dispositivos y también consumen recursos desde la grid.

4.1.8 Grid móvil

Tipo de grid que se enmarca en la categoría de grids accesibles[30] y en la cual, de acuerdo con [22] pueden conectarse usuarios a través de sus dispositivos móviles (teléfonos inteligentes, tabletas, etc.) con dos propósitos: i) obtener acceso a todos los recursos de almacenamiento, procesamiento, ancho de banda, etc., que provee la grid y/o ii) colocar a disposición de los usuarios de la grid sus propios dispositivos y los recursos con los que éstos cuentan; esto permite ampliar las capacidades de poder computacional, almacenamiento y en algunos casos agregando nuevas características gracias a los sensores que poseen este tipo de dispositivos, tales como cámaras, micrófono, GPS, acelerómetro, etc.

Existen principalmente dos modelos de arquitectura para las grids móviles: una configuración tipo ad hoc (**Figura 2**, derecha), en la cual los dispositivos se conectan y gestionan de forma directa usando protocolos de comunicación tipo P2P; y otra configuración tipo proxy (**Figura 2**, izquierda), en la cual un componente no móvil actúa como intermediario para la gestión de la grid y la planificación y distribución de los trabajos entre los dispositivos. El presente trabajo se centra en las grids móviles tipo proxy.

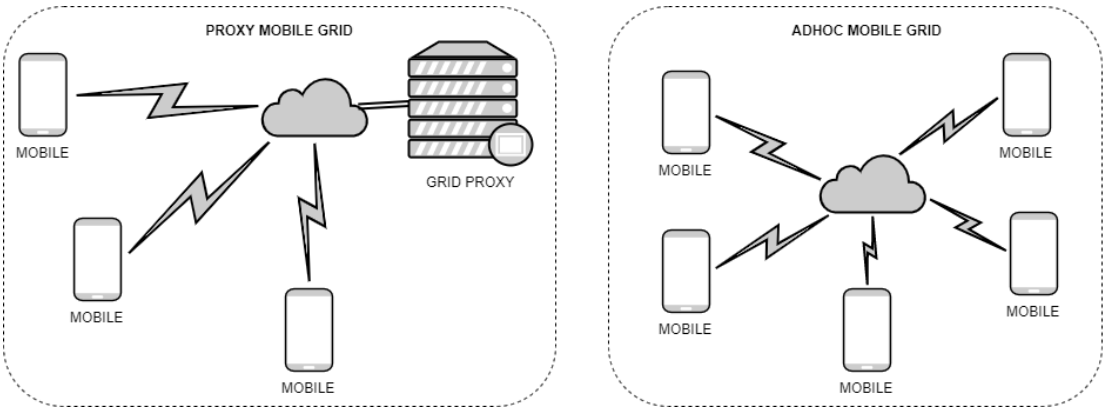


Figura 2 - Tipos de arquitectura en grids móviles

4.1.9 Planificación de trabajos en grids móviles

Durante el proceso de planificación y/o ejecución de los trabajos en grid móviles, las comunicaciones inalámbricas entre el planificador y los recursos móviles pueden verse interrumpidas parcial o totalmente. Además, otras características de estos dispositivos, como el límite de la batería y las capacidades de la memoria RAM pueden hacer que los trabajos enviados no sean finalizados en el tiempo esperado por el servidor.

La confiabilidad y la disponibilidad son aspectos importantes al seleccionar dispositivos de procesamiento para la ejecución de unidades de trabajo bajo restricción de QoS[31]. La disponibilidad se define como la posibilidad de que un usuario pueda usar un sistema de inmediato en un momento especifico y la confiabilidad se refiere a las garantías que tiene el administrador de la grid de que sus tareas terminen a pesar de las fallas en los dispositivos. Las estrategias de planificación implementadas en las grids móviles deben garantizar, en lo posible, estos dos aspectos. En [22] se clasifican las estrategias de planificación para grids móviles en dos grupos: preventivas y reactivas, como se muestra en la **Figura 3**.

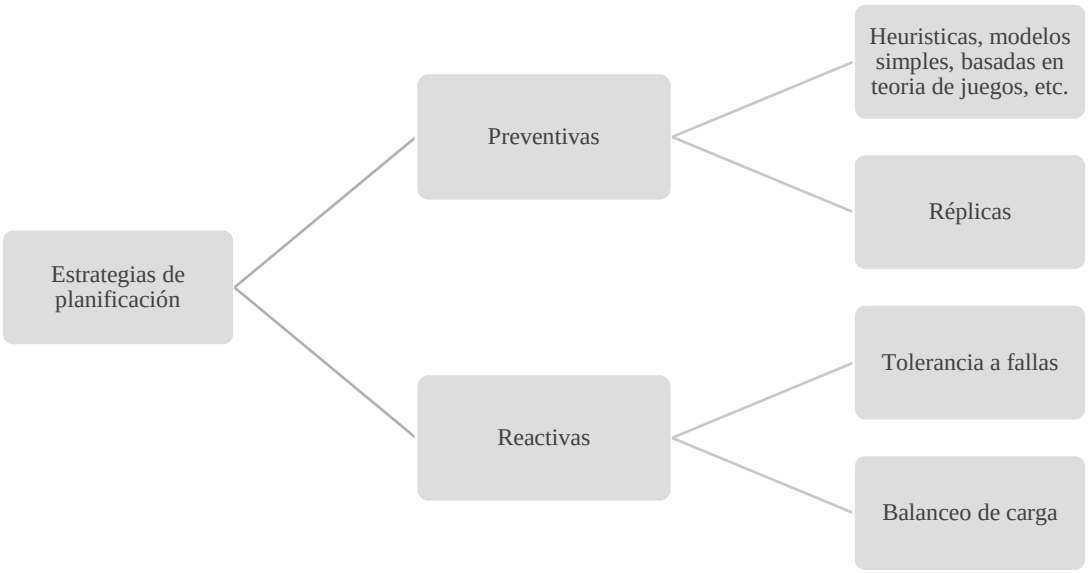


Figura 3 Clasificación de las estrategias de planificación en grids móviles

Las estrategias de planificación preventivas como las descritas en [32][33][34] intentan garantizar la culminación del trabajo, ya sea i) seleccionando el recurso más confiable para ejecutar las tareas mediante el uso de modelos predictivos que usen el estado actual de los dispositivos como parámetros de entrada, técnicas de teoría de juegos y/u otro tipo de heurísticas; o ii) utilizando recursos redundantes en la ejecución (réplicas). Es importante aclarar que, en algunos algoritmos la planificación con réplicas se realiza después de una falla en un nodo de ejecución, en este caso la estrategia se vuelve reactiva. Las estrategias de planificación reactivas actúan después de que las tareas han sido asignadas a los recursos. En este caso, al menos dos escenarios pueden ser posibles: i) reasignación de tareas cuando los recursos fallan o están próximos a fallar y ii) reasignación de tareas para equilibrar la carga de los diferentes nodos y así maximizar la

cantidad de tareas que se pueden completar. Ejemplos de estrategias reactivas pueden encontrarse en [35] y [36].

4.2 BOINC

BOINC es una plataforma de software diseñada para la resolución de problemas mediante computación voluntaria. Fue desarrollada originalmente en la Universidad de California, Berkeley y actualmente existen proyectos usando BOINC en campos como la física, medicina nuclear, climatología y astronomía, entre otros[2]. La herramienta aprovecha la gran capacidad de computo que resulta luego del esfuerzo coordinado de miles de computadoras personales alrededor del mundo cuyos tiempos ociosos son donados por los usuarios. Luego de la aparición y popularización de los teléfonos inteligentes el proyecto sumó una aplicación cliente para Android, con lo cual también es posible integrar dispositivos móviles con este sistema operativo[37].

La arquitectura general de despliegue de BOINC se muestra en la **Figura 4**, en la cual se puede observar los principales componentes que la conforman, a saber: i) los dispositivos de computo aportados por los voluntarios, representados en el diagrama como un teléfono móvil, pero que puede ser también un PC de escritorio, ii) un servidor que mantiene la información del proyecto en archivos de configuración y base de datos y iii) un componente interno dentro de BOINC que se encarga de realizar la planificación de los trabajos hacia los dispositivos, los cuales se comunican con el servidor mediante el intercambio de mensajes a través del protocolo HTTP.

Para usar BOINC como plataforma de computación se debe generar un **proyecto**. Este proyecto pertenece a una organización o grupo de investigación interesado en resolver un problema que requiere gran capacidad de computo mediante el uso de la computación voluntaria o computación en grid. El proyecto está identificado por una URL maestra, la cual sirve como acceso no solo de las aplicaciones cliente, sino como recurso de administración de los trabajos y aplicaciones. Los voluntarios se registran y colaboran con proyectos específicos de acuerdo con sus preferencias.

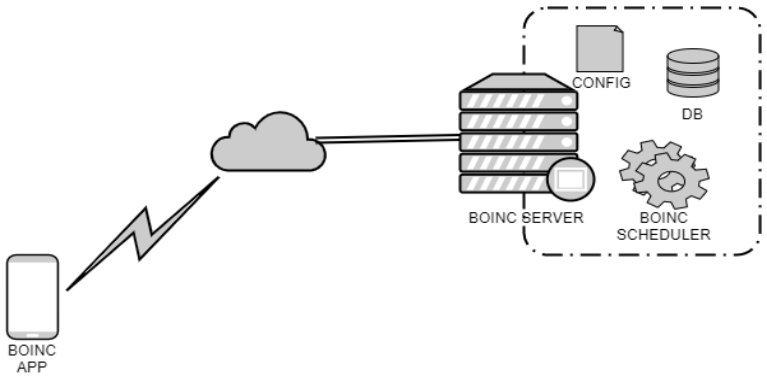


Figura 4 Arquitectura BOINC

Un proyecto puede incluir múltiples aplicaciones[38]. Una aplicación incluye varios programas con su respectiva versión para distintas plataformas, un conjunto de unidades de trabajo (*workunits* en términos de BOINC) y resultados. Una unidad de trabajo describe el cálculo a realizar. Cada unidad de trabajo incluye varios parámetros útiles para el algoritmo de planificación. Estos parámetros son estimaciones de: i) la cantidad de operaciones de punto flotante por segundo

(FLOPS) que realiza la *workunit*, ii) la cantidad máxima de memoria RAM que puede consumir, iii) la cantidad máxima de almacenamiento del dispositivo que puede ser usada y iv) un estimado del tiempo total de ejecución del trabajo. BOINC también puede estimar el tiempo de ejecución de acuerdo con la cantidad de FLOPS del trabajo y la cantidad de operaciones que soporta el procesador del nodo de ejecución.

Cada cálculo realizado genera un resultado, el cual consiste en una referencia a una *workunit* y opcionalmente una lista de referencias a archivos de salida generados por el computo realizado. En algunos casos se crean varias réplicas de una unidad de trabajo determinada y por lo tanto se obtienen varios resultados desde diferentes nodos de ejecución. El uso de réplicas en BOINC se describe en la sección **4.2.2 Réplicas en BOINC**.

Además de la aplicación que se encarga de resolver las unidades de trabajo en los dispositivos móviles, un proyecto debe implementar un proceso generador de trabajos o ***work_generator***, el cual se encargará de generar de forma continua las *workunits* que serán procesadas por el planificador para ser enviadas a los dispositivos. Para la recepción de los trabajos se debe implementar un validador o ***validator***, el cual se encarga de verificar los resultados y marcarlos como válidos o erróneos. Por último, el asimilador o ***asimilator*** es el encargado de usar los archivos y cálculos de respuesta enviados juntos con los resultados que han sido marcados como validos por el validador. En un caso de uso común el asimilador traslada los resultados a otro servidor o componente externo para ser usados en procesos posteriores.

4.2.1 Planificación de trabajos en BOINC

La planificación de tareas en BOINC está asociada a su modelo de computación voluntaria. Existen dos componentes involucrados en la planificación: uno ejecutado en el cliente (o dispositivo voluntario), que se encarga de determinar cuándo y cuántos trabajos nuevos solicitar como máximo al servidor y, en el caso de estar participando en más de un proyecto de forma simultánea, decidir para cuál de ellos realizar solicitudes. El otro componente es el ejecutado en el servidor y se encarga de decidir cuántos trabajos enviar a un determinado cliente de acuerdo con sus características de procesamiento, memoria y almacenamiento; además del tipo de sistema operativo y arquitectura del procesador.

Cuando un voluntario BOINC se encuentra asociado a un proyecto, su rutina de planificación periódicamente realiza solicitudes al planificador de trabajos en el servidor mediante el envío de mensajes en formato XML, los cuales contienen: i) una descripción del dispositivo donde se está ejecutando el cliente y su carga de trabajo actual (*workunits* encoladas y en ejecución); ii) descripción de las *workunits* que se han terminado de ejecutar y los archivos de salida generados y iii) una petición de nuevas unidades de trabajo. Adicionalmente, el planificador ejecutado del lado del cliente se encarga de determinar el orden de ejecución de los trabajos que ya se encuentran descargados localmente. La política de planificación de trabajos local que mantiene el cliente BOINC en cada nodo de ejecución cumple con varios objetivos, entre ellos: maximizar el uso de recursos, cumplir con los tiempos límites de procesamiento de las unidades de trabajo establecidos por el administrador, respetar los requerimientos de los usuarios con respecto al uso de los recursos de sus dispositivos y mantener unidades de trabajo de todos los proyectos en los que se está participando.

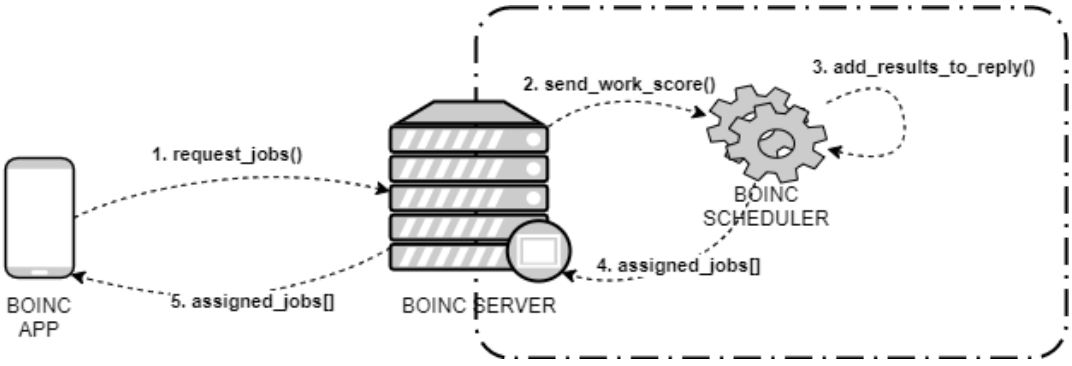


Figura 5 – Planificación de trabajos en BOINC

El planificador de tareas que se encuentra en el servidor responde las peticiones enviadas por los dispositivos de computo. La selección de las unidades de trabajo que se le enviarán a los voluntarios se realiza mediante una política de planificación basada en puntaje. La política se implementa con la función de puntaje *P(N,W)* que busca enviar al nodo de ejecución *N* las unidades de trabajo *W* para las cuales el resultado de la función *P(N,W)* es mayor. La función usada por defecto en BOINC combina parámetros relacionados con[39]: i) la cantidad esperada de operaciones de punto flotante por segundo (FLOPS) de la versión de la aplicación disponible para el dispositivo *N*; ii) requerimientos de RAM y almacenamiento de la aplicación; iii) número de réplicas a enviar y iv) clase de redundancia homogénea a la cual pertenece *N*[40]. La redundancia homogénea y el uso de réplicas se explican en los próximos párrafos.

En la **Figura 5** se resume el proceso de solicitud y entrega de trabajos a los dispositivos voluntarios en BOINC, la rutina **send_work_score()** implementa la función de puntaje descrita anteriormente, la cual usa la función interna **add_results_to_reply()** para seleccionar los trabajos que deben ser entregados al dispositivo que se encuentra realizando la solicitud. El planificador de tareas en su conjunto está implementado en BOINC en el lenguaje de programación C++ y es el componente marcado en el diagrama como **BOINC SCHEDULER**, el cual es reemplazado en BOINC-MGE por otro mecanismo que integra a través de una API de desarrollo estrategias de planificación que resuelven los desafíos de batería y desconexión presentes en las grids móviles.

4.2.2 Réplicas en BOINC

En la computación voluntaria cualquier persona con un dispositivo puede ser participe, esto conlleva a que no se puede asumir que todos ellos son confiables o bien intencionados. Adicionalmente, el hecho de que BOINC es de código abierto y fácilmente accesible en internet, conlleva el riesgo de recibir resultados erróneos por parte de uno o varios de los nodos de ejecución. Para mitigar el riesgo de obtener resultados erróneos, BOINC implementa un mecanismo de réplicas en el cual la misma unidad de trabajo es enviada a varios clientes, y luego, a través de un sistema de *quorum,* se verifica que el resultado retornado por un dispositivo coincida con los resultados provenientes de uno u otros nodos de ejecución para la misma unidad de trabajo. Cada proyecto debe habilitar el uso de este mecanismo indicando la cantidad de réplicas a generar y el número de resultados requeridos en el *quorum* para marcar una *workunit* como finalizada.

Sin embargo, debido a que diferentes tipos de procesadores y/o sistemas operativos pueden retornar diferentes resultados para una misma serie de operaciones, como consecuencia, por

ejemplo, de la utilización de distintas librerías de funciones matemáticas, existe en BOINC la replicación homogénea. La replicación homogénea consiste en clasificar las unidades de trabajo a ejecutar en distintas clases con base en el tipo de procesador y sistema operativo.

Existe, además, la **replicación adaptativa**, la cual consiste en determinar si un dispositivo es confiable o no dependiendo de la cantidad de resultados consecutivos que retorna al servidor de forma satisfactoria y del tiempo que tarda en ejecutar la unidad de trabajo comparado con el tiempo límite de ejecución establecido por el administrador de la grid. Al momento de enviar por primera vez una unidad de trabajo se determina si el dispositivo es confiable y en caso de ser así no se genera ninguna replica de la *workunit*, en caso contrario se generan la cantidad de réplicas indicadas por el administrador. Además, se establece el valor de *quorum* requerido igual al número de réplicas generadas. Se requieren mínimo 10 unidades de trabajo consecutivas sin errores y que el tiempo de ejecución de una unidad de trabajo no sea superior al promedio del resto de tareas finalizadas anteriormente por el dispositivo para marcarlo como confiable. Este es el tipo de replicación recomendado por BOINC debido a que intenta reducir la cantidad de réplicas generadas con el paso del tiempo.

5 Análisis y selección de estrategias de planificación

En la sección **4.1.9 Planificación de trabajos en grids móviles** se clasificaron las estrategias de planificación en reactivas y preventivas, debido a que las estrategias de planificación reactivas actúan una vez los trabajos ya se encuentran en los dispositivos a los cuales fueron asignados, es necesario contar en la grid con mecanismos de migración de trabajos entre nodos. BOINC ofrece a través de su API para desarrollo de aplicaciones[41] funciones para la generación de puntos de control en los dispositivos, de tal forma que se puedan pausar y restaurar tareas enviadas siempre que las aplicaciones que hacen parte de un proyecto así lo implementen. Sin embargo, en BOINC los puntos de control le pertenecen al dispositivo que lo genera, debido a que están asociados con una versión particular de la aplicación y además son almacenados en el mismo dispositivo y no en el servidor BOINC. Debido a esto, no es posible en BOINC implementar estrategias de tipo reactivas sin antes realizar modificaciones importantes tanto en el componente del servidor como en el cliente de ejecución de aplicaciones en los dispositivos; por lo tanto, el alcance del trabajo se limita a la implementación de estrategias de tipo preventivas.

En lo que resta del capítulo se describen las estrategias de tipo preventivas que fueron evaluadas y el mecanismo de selección usado, el cual utiliza una matriz de evaluación basada en criterios de la metodología DAR[24] para toma de decisiones. Al final del proceso, dos estrategias fueron seleccionadas para ser integradas a BOINC, una basada en modelos predictivos que tiene en cuenta el estado de la batería y otra basada en réplicas que brinda una solución al desafío de las desconexiones de los dispositivos.

5.1 Estrategias de planificación revisadas

La **Tabla 1** resume las estrategias revisadas por cada subcategoría dentro de la clasificación de estrategias preventivas. En cada grupo se seleccionó una estrategia para ser integrada a BOINC de acuerdo con los resultados obtenidos luego de aplicar la metodología DAR.

Tabla 1 *Estrategias preventivas evaluadas*

Basadas en modelos	Basadas en réplicas
SEAS [33]	MGRR [42]
EAAC [43]	FIXEDRPL [44]
PHSA [45]	GARS [46]
ERAOA [47]	RLREPL [48]
ECRTS [49]	
BEATA [50]	
PAS-BTA [51]	

En el subgrupo de las estrategias basadas en modelos se tuvieron en cuenta distintas opciones cuyo objetivo fuese el uso del estado de la batería de los dispositivos como variable principal del modelo de predicción planteado.

Se describen solo aquellas estrategias que fueron sometidas a evaluación mediante la metodología DAR, sin embargo, muchas otras fueron consultadas, las cuales se iban descartando debido a que no estaban diseñadas para grids móviles, no tenían en cuenta el estado de la batería

de los dispositivos o no implementaban un mecanismo de réplicas que permitiera resolver el desafío de las desconexiones.

A continuación, se describen todas las estrategias revisadas para cada subcategoría.

5.1.1 Estrategias basadas en modelos

Estas estrategias fueron diseñadas para la planificación de trabajos en dispositivos móviles y, por lo tanto, tienen en cuenta el estado actual de la batería en los nodos de ejecución antes de realizar la planificación. Mediante el uso de algún tipo de modelo predictivo que, o bien intente seleccionar los dispositivos que puedan finalizar las tareas en un tiempo razonable teniendo en cuenta la cantidad de batería actual, y/o generen un menor consumo de energía en los dispositivos de todo el sistema grid al que pertenecen.

5.1.1.1 SEAS – Simple Energy Aware Scheduler[33]

La estrategia SEAS se basa en determinar el tiempo disponible de procesamiento que tiene un dispositivo móvil antes de agotar su batería. Para ello se realizan los siguientes pasos: a) se calcula la tasa de descarga de la batería, tomando inicialmente la cantidad de batería con la que cuenta el dispositivo (*bc*) y la hora en la que se toma dicho registro (*ct*), b) luego, se espera hasta que haya un cambio en el estado de carga de la batería y se vuelve a medir el nuevo nivel de batería registrado (*nbc*) y la hora en la que dicho cambio ocurre (*nct*), c) con estos datos se puede calcular la tasa de descarga (*dr*) de la siguiente forma:

$$dr = \frac{bc - nbc}{nct - ct}$$

Asumiendo que la tasa de descarga es siempre constante, se puede calcular el tiempo restante en el cual el dispositivo estará disponible (*rt*) mediante la siguiente formula:

$$rt = \frac{nbc}{dr}$$

Con base a las estimaciones anteriores y pudiendo conocer cuál es el tiempo total que consume ejecutar una tarea en un dispositivo, se puede determinar la cantidad de trabajos a enviar de tal forma que éstos puedan terminarse antes de producirse la desconexión del dispositivo por falta de batería.

5.1.1.2 EAAC – Energy Aware Admission Control[43]

Esta estrategia se implementó en una grid de distribución de contenido multimedia. En esta grid, los dispositivos móviles actúan como clientes que desean reproducir videos bajo demanda, por lo cual es necesario que envíen de forma periódica información al servidor sobre el estado de su batería. Esta información es usada por los servidores para distribuir los trabajos. De acuerdo con el estado de la batería (el planificador o los servidores) deciden si pueden aceptar o no una petición de un dispositivo para reproducir un vídeo y la calidad de video generado para el mismo.

El propósito de este esquema es asegurarse que no se malgasten recursos tanto en los nodos de procesamiento de videos en el servidor, como en el dispositivo móvil que los reproduce. Esto

pudiera pasar al enviar videos que no puedan ser vistos en su totalidad por los usuarios debido a la inminente desconexión del dispositivo por agotamiento de la batería.

En el momento en el que un usuario móvil decide reproducir un video en *streaming* y la calidad mínima del mismo (Qm), envía una petición al servidor junto con la información actual del estado de su batería. En este momento la estrategia EAAC[43] realiza una proyección en base a datos históricos y a los parámetros de calidad de servicio previamente indicados por el usuario para determinar si el dispositivo puede reproducir el video solicitado en su totalidad antes de quedarse sin batería, en caso negativo la petición es negada y al dispositivo no se le asigna un servidor de *streaming* para que pueda reproducir el video, en el caso positivo se permite la reproducción del video al dispositivo y se monitorea durante todo el tiempo de reproducción para ir actualizando los datos de consumo de batería y la liberación de los recursos asignados en el servidor en caso de ser necesario.

5.1.1.3 PHSA – Power-Aware Hierarchical Scheduling Algorithm[45]

Estrategia de planificación usada en una grid que mezcla recursos fijos y móviles, los cuales se organizan en orden jerárquico. En el primer nivel de la jerarquía se encuentran los recursos fijos con mayor poder de procesamiento y confiabilidad, y en el segundo nivel se agrupan los recursos móviles sobre los cuales se aplica un algoritmo heurístico ***min-min*** [45] para seleccionar los dispositivos que deben ejecutar una serie de trabajos dentro de la grid.

5.1.1.4 ERAOA – Energy Constrained Resource Allocation Optimization Algorithm[47]

En esta estrategia los dispositivos móviles se modelan como entidades que poseen tres recursos para ofrecer a la grid: red para transferencia de datos, poder computacional y batería. Además, cada dispositivo móvil tiene un conjunto de aplicaciones que pueden ser ejecutadas $A = \{A_1, A_2, \dots, A_i\}$ y un conjunto de recursos $R = \{R_1, R_2, \dots, R_i\}$; C_i es la capacidad disponible del recurso R_i. Un dispositivo móvil estima de forma individual la tasa de consumo de energía er_l para ejecutar un conjunto de aplicaciones A, teniendo en cuenta que su límite de consumo de energía es C_l, el cual impone una restricción de la forma $er_l * t_i \leq C_l$, donde t_i es el tiempo necesario para completar una aplicación i.

Cada aplicación por ejecutar es la sumatoria de varias tareas, las cuales pueden ser ejecutadas en cualquiera de los nodos móviles de procesamiento disponibles en la grid, sin embargo, cada traslado de tareas entre los nodos requiere de un gasto energético por transferencia y otro por ejecución del trabajo. La suma total de los gastos de energía por la utilización de los recursos no puede exceder el total disponible por cada nodo móvil.

El algoritmo ERAOA [47] consta de dos etapas: En la primera, cada uno de los nodos móviles estima un costo para cada uno de los tres recursos de los que dispone (energía, ancho de banda y capacidad de CPU); el planificador por su parte, teniendo en cuenta un presupuesto máximo para el consumo de energía y haciendo uso de modelos matemáticos, estima los valores óptimos que deben ser pagados por cada recurso sin importar cual sea el nodo que los ofrece. En la segunda etapa, los dispositivos móviles ajustan los costos de sus recursos a valores óptimos, teniendo en cuenta las restricciones de energía con las que cuenta y el costo actualizado enviado por el planificador en la primera etapa. Este mecanismo de negociación conlleva, de acuerdo con

los principios de teoría de juegos y cooperación, a un valor óptimo para las dos partes y a una disminución generalizada del consumo de energía en toda la grid.

5.1.1.5 ECRTS – Energy-Constrained Scheduling for weakly-hard Real-Time Systems[49]

La estrategia ECRTS plantea un mecanismo de planificación de trabajos en sistemas de tiempo real con limitaciones de energía usando una técnica denominada DVS (Dynamic Voltage Scaling), que consiste en regular la velocidad del procesador de un sistema embebido mediante la aplicación de mayor o menor voltaje. Esta técnica de escalado, tal como lo plantean los autores, permite una mejora considerable en el consumo total de energía de un dispositivo si se aplican en determinados momentos y durante un periodo de tiempo que permita cumplir con las condiciones de tiempo y consumo de energía requeridos.

En Android es posible aplicar este tipo de escalado en la CPU[52], de tal forma que se puede indicar al procesador que trabaje a mayor o menor velocidad, modificando el consumo de energía durante la ejecución de una aplicación, sin embargo, aplicar este tipo de técnicas puede ocasionar que algunas aplicaciones de uso cotidiano como juegos, correo, etc., vean disminuido su desempeño o funcionen de forma errática. Es por esta razón que se suele usar DVS solo en momentos en los que el dispositivo es usado exclusivamente en aplicaciones que requieren este tipo de técnicas.

5.1.1.6 BEATA – Balanced Energy-Aware Task Allocation[50]

Estrategia que plantea un algoritmo para asignación de tareas, las cuales hacen parte de una aplicación y son ejecutadas de forma paralela. Cada aplicación es modelada como un par (T, E) donde $T = \{t_1, t_2, \dots, t_n\}$ representa un conjunto de tareas y E es un conjunto de ejes ponderados y dirigidos que representan la comunicación entre las diferentes tareas de una aplicación. Siendo ECN_j^{active} la tasa de consumo de energía del nodo j mientras se encuentra ejecutando una tarea, y c_i^j el tiempo que tarda el nodo j en ejecutar la tarea i, el total de energía consumida por un nodo se puede representar como $e_i^j = ECN_j^{active} * c_i^j$. En consecuencia, la energía consumida por todos los nodos X asignados para ejecutar una aplicación T es:

$$en^{active}\left(T, X, ECN^{active}\right) = \sum_{j=1}^{m} ECN_j^{active} \sum_{i=1}^{n} x_{ij} * c_i^j$$

Sin embargo, la ecuación anterior no considera el consumo de energía que puedan tener los dispositivos móviles cuando se encuentran en estado de espera, ni tampoco los consumos por transmisión de datos a través del canal de comunicación. Si se agregan estos dos factores y se suma el consumo de energía de los canales de comunicación cuando no se encuentran trasmitiendo ni recibiendo datos, se puede armar un modelo más complejo que represente el consumo total de energía de la grid cuando se ejecuta en un conjunto de nodos X una aplicación T.

$$e\left(T, X, ECN^{active}, ECN^{idle}, ECL^{active}, ECL^{idle}\right) = en\left(T, X, ECN^{active}, ECN^{idle}\right) + el(T, X, ECL^{active} + ECL^{idle})$$

en es la energía consumida por el nodo móvil y el la energía consumida por el canal de comunicación, los cuales son la sumatoria del tiempo activo y el tiempo que permanecen inactivos.

El algoritmo BEATA [50] distribuye las diferentes tareas de las aplicaciones entre los nodos que forman la grid, mientras se reduce al máximo el consumo de energía de todo el sistema distribuido. Antes de minimizar el consumo de energía de una tarea t_i, BEATA organiza todos los nodos disponibles ordenados de mayor a menor tiempo de finalización de la tarea t_i en cada dispositivo. Luego calcula la cantidad de energía necesaria para ejecutar dicha tarea en cada uno de los nodos, para ello se computan, tanto el gasto de energía por ejecución de la tarea propiamente dicha, como el gasto por el paso de mensajes de tareas previas asociadas. Seguidamente, se selecciona el nodo que consuma menos energía mientras se mantiene el tiempo de ejecución en un rango de aceptación (llamado ventana adaptiva de energía) que no genere una degradación en el rendimiento del sistema. Finalmente, se asigna la tarea al nodo, se actualiza su tiempo disponible de procesamiento y se pasa a la asignación de la siguiente tarea usando los mismos pasos anteriores.

5.1.1.7 PAS-BTA – Power Aware Scheduling of Bag of Task Applications[51]

PAS-BTA es una estrategia de planificación de trabajos en un clúster de dispositivos fijos cuya finalidad es reducir el consumo de energía mientras se cumplen con los tiempos de finalización de las tareas. La estrategia utiliza la reducción de la velocidad del procesador del dispositivo debido a la baja aplicación de voltaje, técnica ya explicada anteriormente y conocida como DVS.

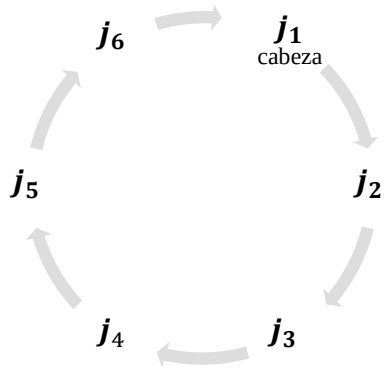
De forma similar a la estrategia ECRTS, la aplicación de técnicas de DVS afecta el comportamiento de todo el dispositivo, con lo cual se pudiera degradar su funcionamiento general sino se aplica de forma adecuada; además, este tipo de estrategia requiere tener más control continuo sobre el dispositivo móvil en el transcurso de la ejecución del trabajo para poder ajustar en tiempo real la velocidad de la CPU.

5.1.2 Estrategias basadas en replicas

La segunda subcategoría en la que se dividen las estrategias preventivas de planificación en grids móviles son aquellas basadas en réplicas. Con este tipo de estrategias se pretende resolver el desafío de las desconexiones de los usuarios, las cuales pueden impedir la finalización de los trabajos en los tiempos requeridos. En la siguiente sección se describen las estrategias que fueron analizadas para ser integradas en BOINC-MGE.

5.1.2.1 MGRR – Mobile Grid Round Robin[42]

Algoritmo que distribuye los trabajos usando el método *round-robin* y una estructura de anillo en la cual se almacenan el listado de tareas que no han sido asignadas a un dispositivo móvil y aquellas que, habiendo sido asignadas aún no han culminado. Inicialmente, el anillo tiene el listado de todas las tareas que se desean ejecutar, las cuales son agregadas al anillo en el mismo orden en que fueron registradas en la grid para su ejecución. La primera tarea registrada se marca como "cabeza" del anillo. La cabeza del anillo indica la siguiente tarea en ser asignada a un dispositivo móvil.



El algoritmo MGRR [42] inicialmente asigna las primeras m tareas a igual número de dispositivos que se encuentren disponibles para ejecución y designa a la tarea j_{m+1} como la nueva cabeza del anillo. Cuando un nuevo dispositivo es registrado en la grid se le asigna la tarea marcada como cabeza y se actualiza la misma a la siguiente tarea en el anillo. Si una tarea ya se encuentra asignada a un nodo, pero aún no ha sido recibida su respuesta y la misma está marcada como cabeza, la tarea es asignada nuevamente, generándose una réplica en un nodo diferente. Este proceso de réplica y asignación tipo *round-robin* es repetido indefinidamente hasta que se recibe el resultado de cada tarea asignada a la grid, momento en el cual son eliminadas del anillo y todas sus réplicas actuales son descartadas, quedando de esta forma los nodos disponibles para tomar nuevas tareas del anillo.

5.1.2.2 FIXED REPL – Fixed Replication[44]

Estrategia de planificación que hace uso de réplicas fijas para asegurar la finalización de los trabajos en la grid móvil. Consiste en indicar por parte del administrador de la grid con antelación el número de réplicas a generar por cada trabajo a ejecutarse en la grid. Con esta información, el planificador de trabajos se encarga de asegurar que las réplicas se distribuyan adecuadamente entre los nodos de la grid. Es una estrategia que mejora la probabilidad de finalización de los trabajos, sin embargo, la forma en que se decide la cantidad de réplicas depende exclusivamente de métodos de prueba y error que no son generalizables a cualquier tipo de grid o de aplicación.

5.1.2.3 GARS – Genetic Algorithm Replication Scheduling[46]

Los algoritmos genéticos son una rama dentro del campo de la inteligencia artificial que, emulando la evolución biológica y su base genético-molecular, pretenden la búsqueda de soluciones optimas a un problema determinado. Estos algoritmos actúan sobre una población inicial de individuos a la cual se aplican acciones aleatorias semejantes a las que actúan en la evolución de los seres vivos (mutaciones y recombinaciones genéticas) y a la selección natural, de acuerdo con un criterio que permite definir cuáles individuos pasan a la siguiente fase evolutiva y cuáles son descartados.

Un algoritmo genético consiste de varias etapas, **inicialización**: donde se genera aleatoriamente la población inicial que está constituida por un conjunto de "cromosomas" que representan posibles soluciones al problema; **evaluación**: durante la cual se aplican funciones de

aptitud para determinar qué tan bueno o malo es un cromosoma en la población; luego se aplican de forma iterativa los siguientes pasos hasta encontrar una solución óptima al problema: **selección**: consiste en determinar cuáles cromosomas serán cruzados en la siguiente generación, teniendo en cuenta los resultados obtenidos en la fase de evaluación; **cruzamiento**: representa la reproducción genética y consiste en combinar dos cromosomas para generar dos nuevos descendientes que posean características de los cromosomas padres; **mutación**: fase en la cual se aplican combinaciones al azar sobre un conjunto de cromosomas para intentar alcanzar zonas del espacio de búsqueda que no estaban en el espectro de la población inicial y finalmente **reemplazo**: fase en la cual se seleccionan los mejores cromosomas para formar la siguiente generación y volver a repetir el proceso.

La estrategia de planificación GARS[46] pretende, mediante el uso de un algoritmo genético, obtener la cantidad optima de réplicas que deben ser generadas para cada uno de los trabajos a ejecutar en una grid móvil. En GARS, la población inicial se forma teniendo en cuenta la confiabilidad de cada uno de los nodos móviles disponibles, por lo tanto, para cada trabajo a ejecutar, se proponen diferentes combinaciones de réplicas entre los nodos disponibles que cumplan con el tiempo de finalización. Un individuo g_i representa una combinación diferente de réplicas para una tarea y sus predecesoras. Para la fase de **evaluación** se usan dos funciones de aptitud, $F_{time}(g_i)$ que mide el tiempo total de finalización de la tarea en el individuo g_i y la función $F_{WR}(g_i)$ que mide la cantidad de recursos desperdiciados en el individuo g_i para ejecutar la tarea. Las fases de **cruzamiento** y **mutación** se llevan a cabo tomando, de forma aleatoria, varios individuos y modificando las combinaciones de réplicas; se usa un esquema simple denominado "*two-point ordered*" o TPO. Finalmente, se aplican de forma iterativa los pasos anteriores y las funciones de evaluación para llegar a una solución óptima de combinaciones de réplicas.

5.1.2.4 RL REPL – Reinforcement Learning Replication[48]

El aprendizaje reforzado o *reinforcement learning* (RL) en inglés, es una técnica de *machine learning* en la cual un agente puede aprender, sin supervisión, a escoger de forma adecuada entre un conjunto de acciones como respuesta a un determinado número de estados. Entre las ventajas de usar RL es que no se requieren datos de entrenamiento y, además, se puede adaptar a ambientes dinámicos.

Un agente que aplica este tipo de técnica basa su entrenamiento en recompensas (retroalimentación positiva) obtenidas por haber tomado una acción adecuada y castigos (retroalimentación negativa) obtenidos cuando la acción tomada no es la correcta.

En [48] se propone un mecanismo basado en RL con el cual se desea determinar, en cualquier momento dado, la cantidad optima de réplicas que deben ser generadas para una unidad de trabajo dentro de la grid, de tal forma que se obtenga la mayor recompensa (probabilidad de que la tarea sea finalizada en el tiempo esperado). La función de recompensa $R_t(s,a)$ en un momento ***s*** para completar una tarea usando ***a*** número de réplicas se define como:

$$R_t(s,a) = \begin{cases} +k - \sigma_t \; si \; t \; ha \; sido \; finalizada \; a \; tiempo \\ -k \; si \; t \; no \; ha \; sido \; finalizada \; a \; tiempo \end{cases}$$

k es un parámetro que permite cuantificar qué tan buena elección fue la cantidad de réplicas generadas para una tarea, $-k$ indica que se ha seleccionado la peor opción de réplicas y $+k$ la mejor opción posible. σ_t por otro lado, es la cantidad proporcional de energía gastada en la ejecución de la tarea por las réplicas generadas.

Para cada tarea a ejecutar se evalúan diferentes cantidades de réplicas para las cuales haya sido posible terminar trabajos anteriores y se toma siempre aquella cuya función de recompensa sea mayor (minimizando la cantidad total de energía consumida).

Sin embargo, dado que la técnica de aprendizaje reforzado consta de dos etapas: una exploratoria, en la cual se buscan nuevas opciones que generen mejores ganancias, y otra de explotación, en la cual se intenta usar al máximo la opción con mejor recompensa, se debe seleccionar de forma aleatoria, en algunos momentos, una opción que permita la finalización de la tarea y cuyo valor de recompensa sea desconocido o menor al optimo actual.

5.2 Criterios de evaluación

Para la evaluación de las estrategias de planificación se hizo uso de una matriz comparativa basada en criterios de la metodología DAR. Los criterios de evaluación están relacionados con la conveniencia, compatibilidad y facilidad de implementación de la estrategia. Para cada criterio se definió un peso para ponderar su importancia. Los criterios y los pesos usados se resumen en la **Tabla 2** y el detalle de cada uno de ellos se describe a continuación.

5.2.1 Independencia del hardware

Debido a que el cliente móvil de la plataforma BOINC es una aplicación en Android y a que dicho sistema operativo restringe el acceso y modificación de componentes de hardware del dispositivo móvil (tales como consumo de energía en métricas diferentes a porcentaje, voltaje del procesador, etc.), se espera que la estrategia seleccionada no tenga una dependencia de métricas y funciones del hardware. Este criterio tiene un peso de 3 puntos y se mide en dos valores: 1 para indicar que la estrategia es independiente del hardware y 0 para indicar que es dependiente.

5.2.2 Compatible con modelo BOINC

Tal como se mencionó en la sección **4.1.8 Grid móvil**, en una grid móvil tipo proxy los nodos que realizan las tareas de computo se comunican con un servidor que actúa como proxy y despacha los trabajos a ejecutar. Este es el tipo de arquitectura implementado por BOINC, en el cual, además, el servidor no asigna los trabajos de forma autónoma a los clientes; por ser un modelo de computación voluntaria, los trabajos se envían a los nodos de computo (móviles o fijos) solo cuando estos se comunican y piden trabajos. El servidor de BOINC tampoco entrega información a los dispositivos sobre otros nodos de computo conectados a la grid, con lo cual las aplicaciones a ejecutar deben ser independientes debido a que no es posible la comunicación entre nodos para la ejecución de tareas en paralelo.

Con este criterio de evaluación se permite identificar y penalizar aquellas estrategias que requieren el uso de alguna característica o arquitectura de grid que no es compatible con el modelo implementado por BOINC, se definen dos valores para evaluar este criterio, el cual tiene un peso de 3 puntos; 0 para aquellas estrategias que para ser implementadas requieren características de

funcionalidad o arquitectura no compatibles con BOINC y 1 para aquellas que pueden ser integradas sin realizar cambios significativos en la plataforma de computación voluntaria BOINC.

5.2.3 Tiempo de convergencia

Uno de los objetivos del presente proyecto es garantizar que los trabajos en ejecución terminen en el tiempo adecuado. Por otro lado, en un modelo de computación voluntaria, los nodos de computo (fijos o móviles) solicitan los trabajos y esperan que la asignación se haga en el menor tiempo posible. Es por esta razón que en la selección de la estrategia se debe privilegiar eficiencia, es decir a aquellas estrategias cuyo algoritmo de planificación logre asignar los trabajos a los dispositivos en el menor tiempo posible. Una estrategia que por ejemplo requiera analizar muchos datos históricos antes de realizar la asignación será penalizada en la evaluación. Se asigna un peso de 1 punto a este criterio y se definen dos valores, 1 para un tiempo de convergencia corto y 0 para un tiempo alto.

5.2.4 Detalles de la implementación

Este criterio de evaluación es usado para favorecer aquellas estrategias que ofrezcan un detalle de la implementación del algoritmo que la soporte, de tal forma que sea más fácil su evaluación preliminar sin necesidad de programarlo en la plataforma BOINC. Adicionalmente, si la estrategia es seleccionada, el contar con un mayor nivel de detalle del algoritmo facilitará su implementación y se evitará la introducción de errores. A este criterio se asigna un peso de 1 punto y consta de dos valores, 1 para aquellas que tengan un algoritmo detallado de implementación y 0 para aquellas que no lo posean.

5.2.5 Estrategia validada

Este criterio de evaluación hace referencia al nivel de evidencias que brinden los autores sobre la efectividad de una determinada estrategia. Se asigna un peso de 1 punto y consta de tres valores, 0 para aquellas que son meramente teóricas, 1 para las que han sido evaluadas e implementadas mediante simulaciones y 2 para las que han sido implementadas y probadas en una plataforma grid real.

Tabla 2. Criterios de evaluación de las estrategias de planificación

Peso	Criterio	Estados
3	Independencia del hardware	INDEPENDIENTE (1) DEPENDIENTE (0)
3	Compatibilidad con modelo BOINC	COMPATIBLE (1) NO COMPATIBLE (0)
1	Tiempo de convergencia	BAJO (1) ALTO (0)
1	Detalle de implementación	DETALLADO (1) NO DETALLADO (0)
1	Validación estrategia	NO VALIDADA (0) PARCIALMENTE VALIDADA (1) VALIDADA (2)

5.3 Evaluación de estrategias

Luego de definir los criterios para la evaluación de las estrategias, se construyó una tabla comparativa ponderando el valor y el peso de cada criterio. Debido a que se pretende seleccionar una estrategia por cada subcategoría de los tipos de planificación en grid móvil, se generaron dos matrices, una por cada subcategoría, conteniendo las estrategias resumidas en la **Tabla 1** y los criterios resumidos en la **Tabla 2**.

La **Tabla 3** muestra los resultados finales de la evaluación, en ella puede observarse que las estrategias que resultaron con mejor calificación son SEAS y RL REPL, la primera corresponde a una estrategia basada en modelos que tiene en cuenta el estado de la batería mientras que la segunda está enfocada en el manejo de réplicas (desconexiones) de forma dinámica usando las desconexiones anteriores de los dispositivos y el consumo de energía como criterios de evaluación. Además, es importante destacar que las dos estrategias seleccionadas no son independientes entre sí, por el contrario, pueden ser usadas de forma conjunta, generando una estrategia unificada que toma la fortaleza de ambas para implementar un mecanismo de planificación adecuado que puede usarse en grids móviles.

El puntaje mostrado en cada criterio corresponde a multiplicar el peso de cada criterio con el valor adecuado, por ejemplo, la estrategia EAAC tiene un puntaje de 2 en el criterio **Validación estrategia** porque el peso de dicho criterio es 1 y se trata de una estrategia que ha sido validada usando una grid real, el cual corresponde al valor 2, por lo tanto, el puntaje final ponderado es 1*2 = 2.

Tabla 3. Evaluación de estrategias de planificación

			Criterios					
			Independencia hardware	**Compatibilidad modelo BOINC**	**Tiempo convergencia**	**Detalle implementación**	**Validación estrategia**	**Calificación Total**
Estrategias de planificación	**Modelos**	*SEAS*	*3*	*3*	*1*	*1*	*1*	***9***
		PHSA	3	3	1	0	1	**8**
		EAAC	3	0	1	1	2	**7**
		ERAOA	3	0	1	1	1	**6**
		BEATA	3	0	1	1	1	**6**
		ECRTS	0	0	1	1	1	**3**
		PAS-BTA	0	0	0	1	1	**2**
	Réplicas	*RLREPL*	*3*	*3*	*1*	*0*	*2*	***9***
		FIXEDRPL	3	3	0	0	0	**6**
		MGRR	3	0	1	1	1	**6**
		GARS	3	0	0	1	0	**4**

El detalle de implementación y la forma como son usadas estas dos estrategias en el proyecto desarrollado es descrito en el capítulo **8. Estrategias de planificación en BOINC-MGE**.

6 Arquitectura y componentes BOINC-MGE

A partir de este capítulo se inicia la ejecución de la Fase 2 del proyecto mediante el uso de la metodología *Adaptive Software Development* (ASD). Se trata de una metodología de tipo ágil con funcionamiento cíclico, que permite reconocer y corregir errores, así como agregar nuevas funcionalidades en un componente software a lo largo de diferentes iteraciones. En este capítulo se presentan los principales componentes que fueron introducidos y modificados en BOINC para generar la extensión BOINC-MGE.

6.1 Arquitectura general

La arquitectura general de despliegue de BOINC-MGE es igual a la arquitectura tradicional de BOINC mostrada en la **Figura 4**, los mismos componentes están presentes en BOINC-MGE, sin embargo, fue necesario realizar modificaciones en cada uno de ellos (archivos de configuración, base de datos, aplicación móvil y planificador de trabajos) para agregar las funcionalidades que permitieran integrar las estrategias de planificación que tengan en cuenta el estado de la batería y las desconexiones de los dispositivos cuando se trata de una grid móvil.

El planificador de trabajos en particular, marcado en la **Figura 6** como BOINC-MGE SCHEDULER fue extendido para agregar un componente extra que permitiera la integración a futuro de nuevas estrategias de planificación.

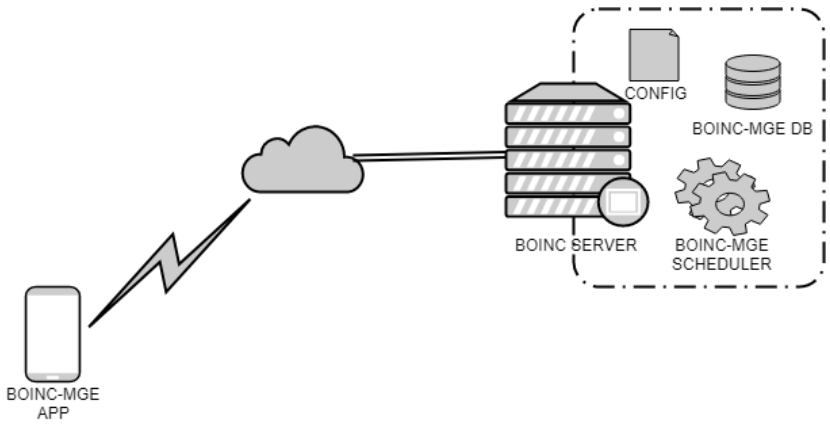


Figura 6 - Arquitectura general de BOINC-MGE

6.2 Componentes BOINC-MGE

En la presente sección se describen los componentes de BOINC que fueron modificados para la extensión BOINC-MGE.

6.2.1 Aplicación cliente Android

La aplicación cliente usada por los usuarios voluntarios que se desean integrar a un proyecto dentro de BOINC y que se ilustra en la **Figura 6** como BOINC-MGE APP, sigue siendo

usada en la extensión BOINC-MGE para agregar un dispositivo a un proyecto BOINC con mayores propiedades de grid móvil. Sin embargo, es necesario contar con una versión modificada de este cliente para Android, la cual agrega una opción de configuración extra que permite a un usuario determinar si desea o no habilitar la extensión para grids móviles, dicha opción, al estar seleccionada, permitirá que el cliente BOINC que se ejecuta en el dispositivo envíe información del estado de la batería y pueda solicitar al planificador (en el servidor) que el envío de trabajos tenga en cuenta el estado de su batería y la tasa de consumo. Como se observa en la **Figura 7**, esta opción está disponible a través del menú de "Preferencias" y por defecto no se encuentra seleccionada. Sin embargo, se debe tener en cuenta que si en el servidor no se encuentra habilitado BOINC-MGE, esta opción no tendrá ningún efecto.

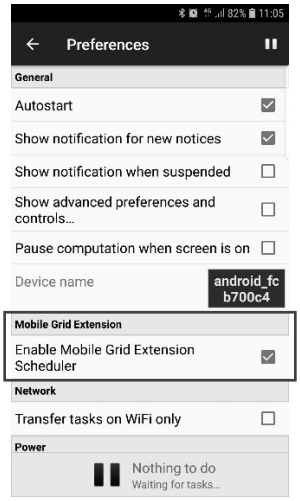


Figura 7 - Opción BOINC-MGE en cliente Android

6.2.2 Servidor BOINC

En BOINC-MGE el componente software del servidor fue modificado para que pudiera interactuar con las estrategias de planificación orientadas a grids móviles. En el código fuente de BOINC se agregaron diferentes estructuras de datos y flujos dentro del circuito de planificación para poder generar la versión del servidor de BOINC-MGE.

Este componente es marcado en la **Figura 6** como BOINC-SERVER y es el encargado de verificar en los archivos de configuración y tablas de base de datos si la extensión MGE está habilitada, y de ser así ejecutará el planificador de trabajos BOINC-MGE SCHEDULER.

6.2.3 Base de datos

Cada proyecto en BOINC genera una base de datos (ilustrada en la **Figura 6** como BOINC-MGE DB), la cual contiene diferentes tablas de información del proyecto, trabajos por ejecutar y en ejecución, usuarios registrados, aplicaciones, estadísticas generales, etc.

Al conjunto de tablas mencionadas se agrega una nueva que mantiene el estado de los dispositivos móviles que conforman la grid, y que tienen habilitada en el cliente Android la opción de planificación para grids móviles. Esta tabla debe ser creada en las bases de datos de todos los proyectos de computación voluntaria que deseen habilitar las funcionalidades de BOINC-MGE.

La nueva tabla **device_status** permite llevar un registro del estado de la batería, así como indicadores del origen de carga del dispositivo (carga mediante USB o corriente eléctrica), conexión mediante una red WiFi y si el usuario se encuentra usando el dispositivo actualmente o no. Esta información puede ser consultada por las nuevas estrategias de planificación usando las funciones de ayuda que expone la API de desarrollo descrita en el capítulo **7. API para integración de algoritmos de planificación en BOINC-MGE**.

Adicionalmente, en la tabla **result** que posee actualmente BOINC y que registra los resultados de las tareas ejecutadas por los nodos de computo, se agrega una serie de campos que permiten registrar el estado de la batería en el dispositivo en el momento en que se le asigna un trabajo y en el momento en el que es finalizado. Esto permite, por ejemplo, medir la cantidad de energía que se consumió en la ejecución de dicha tarea, la cual puede ser usada en la implementación de otras estrategias de planificación haciendo uso de la estructura para acceso a base de datos de BOINC **DB_RESULT**.

6.2.4 Archivos de configuración

Corresponden al componente CONFIG mostrado en la **Figura 6** y hace referencia a los archivos de configuración que se generan cuando se crea un proyecto BOINC, los cuales indican características que deben ser aplicadas al proyecto. En el caso particular de BOINC-MGE se agregaron nuevos parámetros de configuración en el archivo **config.xml**, los cuales determinarán si se usan las estrategias de planificación que tienen en cuenta la batería y desconexiones para asignar trabajos a un dispositivo móvil que solicita tareas al servidor. Estos parámetros actúan en conjunto con la opción de configuración del cliente Android descrita anteriormente, con lo cual tanto el usuario como el administrador del proyecto deben habilitar la extensión MGE para ser efectivamente utilizada. El parámetro **mge_scheduling** para indicar que se debe usar la estrategia basada en modelos y el parámetro **mge_replication** para indicar que se debe usar la estrategia basada en réplicas.

6.2.5 Planificador de trabajos

El planificador de trabajos en el servidor (BOINC-MGE SCHEDULER en la **Figura 6**) es el componente de BOINC sobre el cual se hicieron el mayor número de modificaciones para la generación de la extensión para grids móviles. En él se integran las estrategias de planificación que tienen en cuenta el estado de la batería y las desconexiones seleccionadas en el capítulo anterior. Sin embargo, para dar la posibilidad a futuros trabajos de integrar nuevas estrategias o extender las que serán agregadas en este trabajo, se generó una API de desarrollo que sirve como puente entre el planificador y los algoritmos que implementan las estrategias. El detalle del funcionamiento y estructura de la API de desarrollo es descrito en el capítulo **7. API para integración de algoritmos de planificación en BOINC-MGE** y las estrategias integradas que usan dicha API son descritas en el capítulo **8. Estrategias de planificación en BOINC-MGE**.

7 API para integración de algoritmos de planificación en BOINC-MGE

En este capítulo se presenta el detalle de la API de desarrollo para la integración de estrategias de planificación en BOINC-MGE, la cual permite desarrollar a futuro nuevas estrategias que reemplacen o extiendan las dos seleccionadas para ser integradas en el presente trabajo. La API fue diseñada y desarrollada mediante el ciclo iterativo de la metodología ASD y, además, fue refinada a medida que se iban implementando las estrategias seleccionadas.

7.1 Arquitectura general API

En la **Figura 8** se observa una versión modificada del proceso de planificación de BOINC presentado anteriormente en la **Figura 5**, mostrando los nuevos componentes internos del planificador que hacen parte de BOINC-MGE. La API de desarrollo ilustrada en el diagrama como **BOINC-MGE API** representa el puente entre el nuevo planificador y las estrategias que implementan la lógica que lo soportan. Las estrategias de planificación implementan a través de la API dos rutinas que son invocadas por BOINC-MGE para determinar la forma en que se deben distribuir los trabajos disponibles hacia un dispositivo móvil (*3. send_work_host*) y la cantidad de réplicas que deben ser generadas para un trabajo en particular la primera vez que se asigna a un dispositivo (*5. calc_workunit_replicas*).

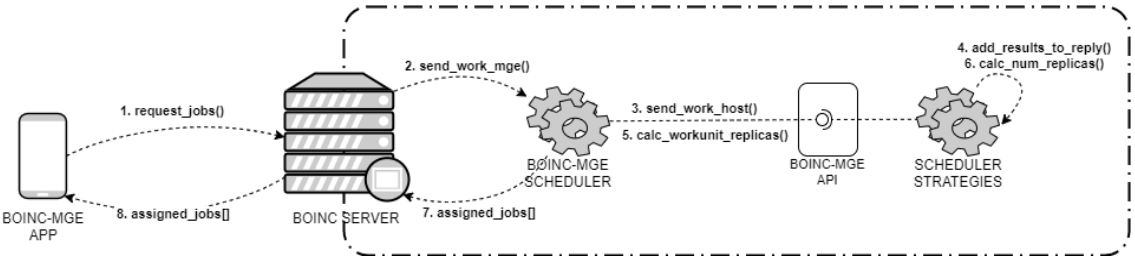


Figura 8 – Planificación de trabajos en BOINC-MGE

El componente **SCHEDULER STRATEGIES** representa una instancia particular de las estrategias de planificación para grids móviles, la cual implementa las dos rutinas mínimas de integración y, haciendo uso de rutinas de ayuda internas de la API (como *4. add_results_to_reply*) realiza la asignación de trabajos y cálculo de réplicas a generar.

BOINC-MGE integra por defecto dos estrategias de planificación, las cuales son descritas en el capítulo **8. Estrategias de planificación en BOINC-MGE**, sin embargo, el desarrollo de la API propuesta en el presente capítulo permite a futuros proyectos reemplazarlas con poco esfuerzo.

7.2 Descripción de la API

La API desarrollada consta de dos grupos de métodos, el primer grupo llamado “métodos de implementación” corresponden a la interfaz que debe ser implementada por el programador

para que BOINC-MGE pueda interpretarlos como parte de una estrategia de planificación; el segundo grupo, denominado "métodos de ayuda" contiene métodos que ya se encuentran implementados en la API y sirven de ayuda para que el desarrollador de la estrategia de planificación pueda integrarse con BOINC sin conocer demasiados detalles de su funcionamiento interno.

La API se encuentra implementada en C++ por ser el lenguaje de programación usado por BOINC para el desarrollo de sus componentes de planificación internos, razón por la cual las estrategias de planificación que se deseen desarrollar deben ser escritas también en este lenguaje. Los métodos que conforman la API se describen a continuación.

7.2.1 Métodos de implementación

En esta sección se describe el grupo de métodos que deben ser implementados por la estrategia especifica que se desea desarrollar. Cuando se encuentra habilitada la extensión MGE el planificador de BOINC llama a estos métodos cada vez que un nodo móvil solicita trabajos al servidor, en estos métodos se debe agregar la lógica para la distribución de los trabajos y, en caso de ser necesario, el cálculo de la cantidad de réplicas a generar por cada unidad de trabajo. Cada método corresponde a uno de los dos tipos de estrategias dentro del grupo de las preventivas mostradas en la sección **4.1.8 Planificación de trabajos en grids móviles**, es decir una basada en modelos predictivos y otra basada en replicas.

- **send_work_host**

  Corresponde al método que implementa el algoritmo que soporta la estrategia de planificación basada en modelos predictivos. Este método es invocado por el planificador y en él se reciben como parámetros un listado de trabajos disponibles en el servidor para ser enviados y una estructura interna de BOINC que contiene información sobre el dispositivo que está realizando la petición de trabajos. El método debe implementar la lógica para decidir cuáles de los trabajos dentro de la lista recibida deben ser asignados al dispositivo móvil.

- **calc_workunit_replicas**
  Método que debe ser implementado para agregar la lógica que soporta la estrategia basada en replicas. Es ejecutado por el planificador de BOINC-MGE cada vez que el método anterior decide asignar un trabajo a un dispositivo y siempre y cuando se haya configurado el proyecto para usar replicas. Cuando el administrador de la grid envía trabajos a ejecutar y no indica una cantidad de réplicas fijas a generar, este mecanismo es aplicado por BOINC-MGE para calcular el número de réplicas en forma dinámica. Además, para mantener compatibilidad con el modelo de réplicas de BOINC como protección contra usuarios maliciosos, también se debe indicar en este método cuantos resultados son necesarios para marcar un trabajo como finalizado.

7.2.2 Métodos de ayuda

Estos métodos ya se encuentran implementados en la API, y están disponibles como herramientas de ayuda para consultar sobre características o realizar acciones relacionadas con la planificación de trabajos sin conocer el detalle del funcionamiento e implementación de BOINC.

- **get_best_app_version**

  Método que retorna la mejor versión de aplicación disponible para una unidad de trabajo. Es decir, aquella con la versión más reciente y que puede ser ejecutada en el dispositivo, teniendo en cuenta sus características de sistema operativo, espacio de almacenamiento disponible, RAM, etc.

- **estimate_workunit_duration**

  Método que permite calcular el tiempo total teórico que tardaría en ejecutarse un trabajo si se usa una determinada versión de aplicación.

- **avg_turnaround_time**

  Método que permite obtener el tiempo promedio real que tarda el dispositivo en finalizar un trabajo, desde que es entregado por el planificador hasta que es retornado por el dispositivo.

- **add_result_to_reply**

  Método usado para indicarle al planificador de BOINC que un trabajo debe ser entregado al nodo móvil que se encuentra realizando la petición actual. El planificador de BOINC-MGE podrá denegar esta petición de forma inmediata si se está intentando entregar más trabajo que la capacidad de computo, disco y/o memoria RAM soportada por el nodo móvil o si se está superando un límite de configuración establecido a nivel de usuario o de administrador del proyecto.

- **get_last_device_status**

  Permite obtener información específica del dispositivo móvil que se encuentra realizando la petición al planificador. Esta información es retornada en una estructura que contiene datos relacionados con su capacidad de batería disponible, la temperatura de la batería, la forma como se encuentra conectado el dispositivo a la red (móvil o Wi-Fi), si el dispositivo se encuentra en proceso de carga y la forma como se está cargando (USB o corriente eléctrica).

- **save_mge_sched_data**
  Método que permite guardar datos relacionados con el dispositivo móvil que se encuentra realizando la petición al planificador, y que son relevantes para la estrategia de planificación que se está implementando. Este método puede ser usado para guardar información persistente en la base de datos entre diferentes peticiones al planificador.
- **get_mge_sched_data**
  Método que permite recuperar los datos previamente guardados con el método save_mge_sched_data.

8 Estrategias de planificación en BOINC-MGE

Este capítulo presenta el detalle de los algoritmos de las estrategias de planificación integradas en BOINC-MGE seleccionadas en el capítulo **5. Análisis y selección de estrategias de planificación.** Para cada una de las estrategias se describe en forma general el algoritmo que las sustenta y la forma como fueron implementadas haciendo uso de los métodos de la API descrita en el capítulo **7. API para integración de algoritmos de planificación en BOINC-MGE.** En este capítulo se finaliza el desarrollo de la Fase 2 del proyecto.

8.1 Estrategia de planificación basada en modelos predictivos

Como ya se mencionó anteriormente la estrategia de planificación basada en modelos seleccionada luego de realizar una evaluación de diferentes alternativas fue el algoritmo SEAS[33], el cual realiza la planificación de los trabajos hacia los dispositivos teniendo en cuenta el estado de la batería y su tasa promedio de descarga. A continuación, se describe el algoritmo y la forma como fue implementado usando la API de desarrollo.

8.1.1 Descripción algoritmo estrategia SEAS

Luego de haberse registrado al proyecto BOINC y haber habilitado el uso de la extensión MGE en el cliente Android, el proceso de solicitud y asignación de trabajos a un dispositivo sigue el siguiente proceso:

1. El dispositivo recolecta información del porcentaje de batería disponible actualmente y la agrega en la petición de trabajos que es enviada al servidor BOINC.
2. El servidor BOINC recibe la petición del dispositivo y al verificar que está habilitado BOINC-MGE reenvía la petición al planificador de la extensión para grids móviles.
3. El planificador de BOINC-MGE guarda la información del estado de la batería enviada por el dispositivo en la variable **CurrentBatteryPct** y la hora en que fue recibida la información de batería en la variable **LastBatteryUpdateTime.**
4. Si es la primera vez que se están enviando trabajos al dispositivo se registra en la variable **StartTime** la hora actual, la cual será usada más adelante para cálculos posteriores.
5. El planificador envía tantos trabajos al dispositivo como CPUs disponibles para procesamiento tenga.
6. El dispositivo recibe y ejecuta los trabajos enviados por el planificador.
7. Al finalizar los trabajos ejecutados el dispositivo envía al planificador una nueva petición de trabajos informando la finalización de los trabajos anteriores y el porcentaje actual de batería disponible.
8. El servidor BOINC recibe la petición del dispositivo y la reenvía a BOINC-ME para que le distribuya nuevos trabajos.
9. El planificador de BOINC-MGE obtiene la información de la batería enviada por el dispositivo y la compara con la guardada anteriormente en la variable **CurrentBatteryPct**, pudiendo generarse uno de los siguientes escenarios:

a. El dispositivo reporta un menor porcentaje de batería disponible con respecto al anterior guardado por el planificador, con lo cual se calcula su tasa de descarga actual.

$$LastBatteryPct = CurrentBatteryPct$$

$$CurrentBatteryPct = \%\ Bateria\ Actual\ dispositivo$$

$$BatteryDischargeRate = \frac{CurrentTime - LastBatteryUpdateTime}{LastBatteryPct - CurrentBatteryPct} \quad (1)$$

Con la tasa de descarga, se calcula el tiempo que tiene el dispositivo disponible antes de quedarse sin batería.

$$AvailableUpTime = CurrentTime - StartTime + CurrentBatteryPct * BatteryDischargeRate \quad (2)$$

El tiempo disponible antes de quedarse sin batería se promedia con mediciones anteriores y se almacena en la variable **AvgPrevUpTimes** usando la siguiente formula. La cantidad de mediciones usadas para calcular el promedio se almacena en la variable **Samples**.

$$Samples = Samples + 1$$

$$AvgPrevUpTimes = AvgPrevUpTimes + \frac{AvailableUpTime - AvgPrevUpTimes}{Samples} \quad (3)$$

Con el promedio de las estimaciones anteriores se calcula un nuevo valor del tiempo que tiene el dispositivo antes de quedarse sin batería de la siguiente forma.

$$NewAvailableUpTime = AvgPrevUpTimes - (CurrentTime - StartTime) \quad (4)$$

Con el valor del tiempo disponible por el dispositivo calculado en la ecuación 4 se asignan trabajos al dispositivo cuya suma de tiempo de ejecución no supere el valor de **NewAvailableUpTime**, teniendo en cuenta que el dispositivo ejecuta tantos trabajos en paralelo como cantidad de CPUs tenga. Es decir, si el dispositivo tiene 2 CPUs disponibles para cómputo y un tiempo disponible estimado de 35 minutos, el planificador

puede asignar hasta un máximo de 4 unidades de trabajo cuyo tiempo de ejecución individual no supere los 15 minutos cada una.

El algoritmo continúa en el paso 6.

b. El dispositivo reporta el mismo porcentaje de batería disponible con respecto al anterior guardado por el planificador. Si ya existe una estimación promedio de mediciones anteriores (**AvgPrevUpTimes**) se vuelve a calcular el tiempo disponible del dispositivo usando la ecuación 4 y se asignan trabajos al dispositivo hasta cubrir el tiempo disponible. Si por el contrario el planificador no tiene información previa de la batería se continúa al paso 5 del algoritmo.

c. Si el dispositivo reporta mayor porcentaje de batería disponible con respecto al anterior guardado por el planificador se asume que el dispositivo se está cargando actualmente, con lo cual se usa la última estimación promedio calculada (**AvgPrevUpTimes**) para calcular el tiempo disponible y asignar trabajos al dispositivo, luego se continua al paso 6.

8.1.2 Implementación algoritmo estrategia SEAS

Para la implementación del algoritmo de la estrategia SEAS descrito en la sección anterior se hizo uso de la API de desarrollo y algunos métodos auxiliares. A continuación, se describen los principales métodos e implementaciones hechas con respecto a los pasos del algoritmo.

- El algoritmo se encuentra implementado en el archivo **boinc-mge/sched/ sched_seas_rlrepl.cpp** y el método **send_work_host** de acuerdo con lo requerido por la API de desarrollo.
- La información de la batería del dispositivo que es recolectada por el algoritmo de planificación descrita en el paso 3 es tomada de la estructura **SCHEDULER_REQUEST*** que viene como parámetro en el método **send_work_host**. Específicamente de la variable **battery_charge_pct**.
- Los valores calculados en el paso 9 que deben ser almacenados por el planificador para ser usados en cálculos posteriores son guardados usando el método de ayuda de la API **save_mge_sched_data**. De igual forma, cuando se requiere actualizar los cálculos usando datos anteriores, se recuperan los datos previos haciendo uso del método de la API **get_mge_sched_data**.
- Una vez calculado el tiempo que tiene disponible el dispositivo antes de quedarse sin batería es necesario conocer cuántos trabajos pueden ser asignados. Para ello se debe saber el tiempo de ejecución de una unidad de trabajo en el dispositivo, con lo cual se usan los métodos de ayuda de la API **estimate_workunit_duration** cuando aún no se ha reportado ningún resultado por parte del dispositivo (con lo cual se usa un valor estimado) y **avg_turnaround_time** cuando ya hay datos reales de tiempo de ejecución. El método **avg_turnaround_time** de la API retorna un valor 0 cuando aún no se ha podido realizar un cálculo real del tiempo de ejecución

de una unidad de trabajo en un dispositivo, con lo cual en ese caso se debe usar el tiempo estimado retornado por el método **estimate_workunit_duration**.

- Una vez se han determinado la cantidad de trabajos que pueden ser enviados al dispositivo de acuerdo con el tiempo que tiene disponible se realiza la asignación haciendo uso del método de ayuda **add_result_to_reply** de la API.

8.2 Estrategia de planificación basada réplicas

Para la estrategia de planificación basada en réplicas se seleccionó el algoritmo planteado en la estrategia RL REPL[48], correspondiente a un algoritmo de *Reinforcement Learning* que pretende obtener la cantidad de réplicas optima que asegure la finalización de los trabajos en escenarios de desconexión y a la vez genere el menor consumo de energía en los dispositivos de la grid, esto lo hace mediante la aplicación de una función de recompensa que penaliza o premia las decisiones hechas en cálculos de réplicas anteriores. Sin embargo, con el fin de aprovechar la información de consumo de energía calculada en la estrategia basada en modelos predictivos, se incorporó al algoritmo RL REPL el uso de dichas métricas, de tal manera que los cálculos realizados se encuentren más alineados con las tasas de consumo actuales de los dispositivos conectados a la grid.

A continuación, se describe en forma general el algoritmo que sustenta la estrategia planteada y su implementación usando la API de desarrollo de BOINC-MGE y la información proporcionada por el algoritmo SEAS en la estrategia basada en modelos.

8.2.1 Descripción algoritmo estrategia RL REPL

La estrategia de planificación basada en replicas se ejecuta al momento de asignar un trabajo por primera vez a un dispositivo siempre y cuando se tenga habilitado el uso de réplicas en BOINC-MGE. Las siguientes condiciones son tenidas en cuenta siempre que se ejecute este algoritmo:

- El número de réplicas que se pretende calcular siempre estará limitado a un valor entre 1 y 3. Esto debido a que se asume que en un entorno de grid móvil no será necesario un número mayor de réplicas para la finalización de un trabajo. Este valor sin embargo es arbitrario y puede ser modificado en un proyecto de grid real acorde a métricas históricas que se hayan recopilado en proyectos anteriores.
- Como se mencionó en la descripción del algoritmo RL REPL la técnica de *Reinforcement Learning* consta de dos fases, una exploratoria en la cual la cantidad de réplicas se calcula de forma aleatoria entre los valores establecidos (1 a 3) y otra fase de explotación en la cual se calcula el número de réplicas que genere un menor consumo de energía y al mismo tiempo permita cumplir con los tiempos de finalización requeridos. Cada vez que se ejecute el algoritmo se debe decidir en cuál de las dos fases se va a calcular el número de réplicas a generar. Siguiendo las recomendaciones indicadas en el algoritmo RL REPL se ejecuta la estrategia exploratoria un 20% de las veces y la de explotación un 80% de las veces que se ejecuta el algoritmo. La razón por la cual la fase de explotación tiene un mayor porcentaje de ejecución es porque se pretende que la herramienta genere la mayor parte del tiempo la cantidad optima de réplicas.

Cuando BOINC-MGE solicita la ejecución del algoritmo para el cálculo del número de réplicas a generar se siguen los siguientes pasos.

1. Se determina la fase en la que se va a ejecutar el algoritmo (de exploración o de explotación). Para ello se genera un número aleatorio entre 1 y 100 y si el número generado es menor o igual a 20 se ingresa a la fase de exploración, en caso contrario se ingresa a la fase de explotación. Se describen los pasos para cada fase a continuación:
    a. En la fase de **exploración** se determina el número de réplicas generando un número aleatorio entre 1 y 3.
    b. En la fase de **explotación** el número de réplicas se calcula de acuerdo con los siguientes escenarios.
        i. Si previamente no se han generado replicas para ningún trabajo no es posible ingresar a la fase de explotación por no contar con datos de consumo histórico, con lo cual se pasa a una fase de exploración y se ejecuta el paso **a**.
        ii. Si se han generado replicas, pero ninguna de ellas ha sido marcada como finalizada por el servidor, se verifica el tiempo transcurrido desde la asignación para cada una de las réplicas generadas y se verifica si se ha excedido el tiempo límite necesario, en caso positivo se marca este número de réplicas como **mala** y se asigna el peor valor de recompensa a esta selección.
        iii. Si se han generado réplicas que ya han sido finalizadas se calcula la cantidad de energía total desperdiciada en la ejecución de todas ellas y se marca el número de réplicas como **buena** o **mala** de acuerdo a si la unidad de trabajo pudo ser finalizada antes del tiempo límite establecido. Ejemplo, para una cantidad de réplicas igual a 2, se buscan las dos instancias del trabajo generadas y se suma la energía desperdiciada por cada nodo que la ejecutó, suponiendo que el nodo A gastó 2% de batería ejecutando una de las réplicas y el nodo B gastó 3% ejecutando la otra replica generada, el total de energía desperdiciada es de 5%. Además, si los dos o uno de los dos nodos pudo finalizar la unidad de trabajo a tiempo se marca el número de réplicas 2 como **buena**. Se hace el mismo ejercicio para la cantidad de réplicas 1 y 3. En caso de que para una de las réplicas no se haya reportado energía total consumida, por ejemplo, porque nunca se finalizó la tarea, se usa la tasa de descarga promedio calculada en la estrategia SEAS para generar un cálculo estimado de batería desperdiciada usando como referencia los datos de consumo del dispositivo que se encuentra realizando la petición de trabajos. Al ejecutar este paso se genera internamente una tabla como la siguiente:

Número de replicas	Energía desperdiciada	Buena réplica
1	3%	NO

2	5%	SI
3	6%	SI

iv. Para cada número de réplicas se calcula el valor de retorno de la función de recompensa de forma proporcional al total de energía desperdiciada para finalizar la tarea. El valor de recompensa varía entre un valor entre 0 y 10 para las réplicas marcadas como **buena**, y un valor constante de -10 para las réplicas marcadas como **mala**.

v. Luego de contar con el listado de recompensas para cada número de réplicas, se usa como número de réplicas a generar aquella que tenga el retorno de recompensa más alto. En la tabla de ejemplo anterior, la cantidad de réplicas a generar serian 2.

2. Luego de contar con la cantidad de réplicas a generar, se indica siempre que el número de resultados positivos requeridos para marcar una unidad de trabajo como finalizada (también llamado *quorum* en BOINC) es 1. Es decir que no importa la cantidad de réplicas que se generen en el paso anterior, solo será necesario que una de ellas finalice satisfactoriamente para marcar el trabajo como finalizado. No obstante, cuando los otros dispositivos reporten la finalización de una unidad de trabajo que ya ha sido marcada como finalizada, se registrará en la base de datos para ser usado en cálculos posteriores en fase de explotación.
3. El planificador de BOINC-MGE luego de recibir la cantidad de réplicas a generar, entrega una de ellas al dispositivo actual y el resto (de existir) serán entregadas a dispositivos que realicen peticiones al planificador posteriormente.

8.2.2 Implementación algoritmo estrategia RL REPL

La implementación del algoritmo descrito se hizo haciendo uso de la API de desarrollo de BOINC-MGE, la cual especifica que se debe escribir el método **calc_work_unit_replicas** para el cálculo del número de réplicas a generar. A continuación, se describen los métodos usados con respecto a los pasos del algoritmo descrito en la sección anterior.

- Para determinar el tipo de fase en la cual se ejecuta el algoritmo, se usó la función de generación de números aleatorios **rand** del estándar C.
- Para consultar las unidades de trabajo finalizadas por replicas anteriores se usó la estructura **DB_WORKUNIT** de BOINC, la cual permite realizar consultas sobre la base de datos del proyecto BOINC sobre el cual está solicitando trabajos el dispositivo. Para cada unidad de trabajo se consultan los resultados correspondientes a cada replica generada haciendo uso de la estructura de BOINC **DB_RESULT**, la cual permite obtener información sobre la batería consumida en su ejecución y el tiempo total que tardó la unidad de trabajo en ser finalizada, desde que fue asignada a un dispositivo móvil hasta que fue reportado su resultado.
- Para consultar el consumo promedio de batería calculado por el algoritmo SEAS de la estrategia basada en modelos se hizo uso del método de ayuda de la API de desarrollo **get_mge_sched_data**.

9 Pruebas y evaluación de resultados

En este capítulo se describe el procedimiento llevado a cabo para validar en forma experimental las estrategias de planificación implementadas y en general la extensión BOINC-MGE para grids móviles. El capítulo consta de cuatro secciones principales, en la primera se describe un proyecto BOINC creado y configurado para ejecutar las pruebas de las estrategias; en la segunda sección se describe el diseño experimental, la realización de las pruebas y el análisis de los resultados obtenidos al probar la estrategia SEAS basada en modelos, en la tercera sección se describe el conjunto de pruebas y análisis de los resultados obtenidos al validar la estrategia basada en réplicas RL REPL, y, finalmente en la cuarta sección se muestran los resultados de las mediciones realizadas en el servidor para medir el impacto en el consumo de recursos debido al uso de las estrategias de planificación de BOINC-MGE.

9.1 Proyecto grid para el procesamiento paralelo de imágenes

Teniendo en cuenta que el proyecto macro que dio origen al presente trabajo consiste en la utilización de dispositivos móviles para el procesamiento de imágenes médicas, se decidió implementar un proyecto en BOINC que permita realizar un procesamiento sencillo de imágenes, el cual ayudará a evaluar de mejor forma las estrategias implementadas en el contexto de una grid móvil.

El proyecto es una simulación de procesamiento de imágenes de gran tamaño, las cuales serán divididas en imágenes más pequeñas que podrán ser ejecutadas por los dispositivos móviles en forma simultánea. BOINC requiere que se implementen al menos 3 componentes para la generación de un proyecto; el **generador de trabajos**, encargado de generar en la grid las unidades de trabajo a ejecutar, subdividiendo una imagen en porciones más pequeñas que puedan ser enviadas a los dispositivos; la aplicación o **benchmark** que se ejecutará en los dispositivos móviles y se encargará fundamentalmente de detectar los bordes y simular procesamientos sobre la imagen, y el **asimilador** que se ejecuta en el servidor y se encarga de recibir y validar los resultados de los trabajos enviados por lo dispositivos.

Cada uno de estos componentes es descrito a continuación y el código fuente se encuentra disponible como un repositorio independiente a BOINC-MGE en la plataforma **github.com.**

9.1.1 Generador de trabajos *cannyedge_work_generator*

El generador de trabajos es un proceso que corre en el servidor de BOINC de forma constante y se encarga de tomar imágenes desde un directorio específico al cual se han agregado las imágenes que se desean procesar. El generador de trabajos divide la imagen en porciones más pequeñas que puedan ser distribuidas a los dispositivos. Tal como se muestra en la **Figura 9** la imagen es rebanada y por cada porción se genera una unidad de trabajo asociada. Debido a que la imagen es cargada y procesada por el dispositivo usando la memoria RAM se decidió generar sub-imágenes de tamaño no mayor a 6MB.

El proceso **cannyedge_work_generator** es registrado en el proyecto BOINC para generar trabajos a partir de una imagen. La imagen con extensión .bmp debe copiarse en el directorio ***projectdir*/tmp/source_dir/**.

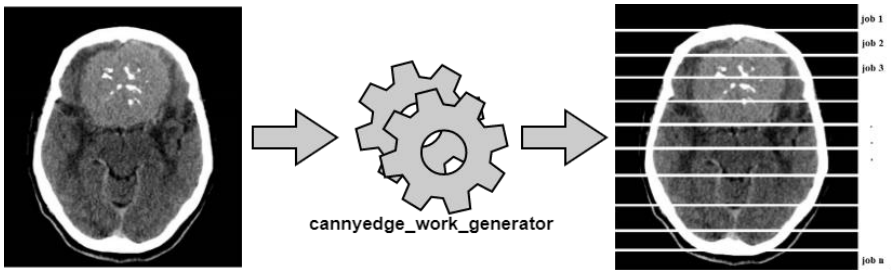


Figura 9 - Generación de trabajos por división de imagen

9.1.2 Benchmark para el procesamiento de la imagen

Para procesar las unidades de trabajo generadas se construyó una aplicación para el procesamiento de una imagen, la cual inicialmente aplica el algoritmo de detección de bordes de Canny[53] y luego realiza otras operaciones de uso de CPU que simulan un procesamiento posterior sobre la imagen, de tal forma que el tiempo total de ejecución de una unidad de trabajo esté por encima de los 10 minutos.

El algoritmo Canny mediante una serie de transformadas y filtros permite aislar los bordes en una imagen. Este tipo de algoritmo es de mucha importancia en el proceso de segmentación de imágenes, ya que ayuda a obtener información relevante en una imagen para el diagnóstico, o sirven como paso inicial para la aplicación de técnicas posteriores.

La aplicación fue desarrollada usando la API de desarrollo de aplicaciones de BOINC y dos librerías externas; CannyEdgeDetector[54], la cual es una implementación libre del algoritmo propiamente dicho y la librería bmp[55], usada para la manipulación y generación de archivos de mapa de bits. La aplicación al ejecutarse recibe la ubicación de un archivo en formato .bmp que contiene una porción de la imagen original y, al finalizar genera una imagen con el mismo formato en blanco y negro con los bordes de la imagen resaltados en blanco, tal y como lo muestra la **Figura 10**. Sin embargo, debido a la relativamente alta capacidad de computo de los dispositivos móviles usados para la ejecución de las pruebas se agregó a la aplicación una porción adicional de procesamiento de CPU que simula otras operaciones sobre la imagen, de tal forma que el tiempo total de procesamiento de una unidad de trabajo esté por encima de los 10 minutos y se encuentre más alineado con tiempos de ejecución que justifiquen el uso de una grid móvil para la paralelización del procesamiento.

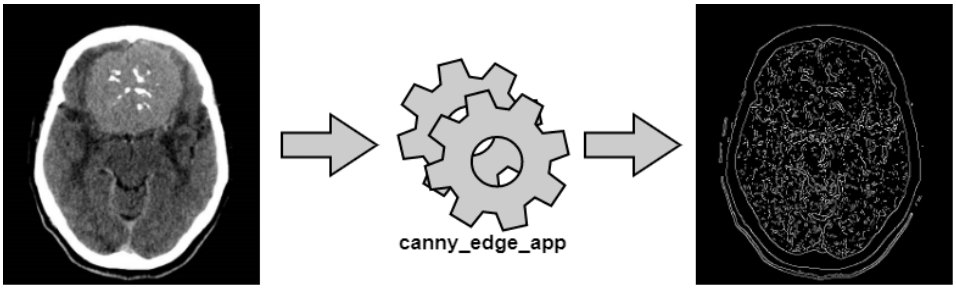


Figura 10 - Transformación de imagen con canny_edge_app

Debido a que la aplicación debía ser ejecutada en dispositivos móviles, se tuvo que compilar para la arquitectura ARM realizando un proceso de compilación cruzada desde una máquina de desarrollo corriendo el sistema operativo Linux Debian 9.7.

9.1.3 Asimilador de resultados *cannyedge_assimilator*

Finalmente, se generó una aplicación de tipo "asimilador", el cual dentro del modelo BOINC es el encargado de verificar los resultados retornados por los nodos móviles una vez que han sido marcados como validos por parte del servidor. Debido a que se trata de una aplicación de ejemplo y no se pretende realizar un procesamiento posterior a las imágenes segmentadas, este componente también se encarga de eliminar las imágenes y liberar espacio de disco en el servidor para evitar el agotamiento de recursos mientras se procesan de forma continua muchas imágenes de gran tamaño.

9.2 Evaluación de estrategia SEAS basada en modelos

Para la evaluación y validación de la estrategia SEAS se decidió llevar a cabo un protocolo experimental de tipo factorial $2^k r$. Este tipo de experimentos ampliamente descritos en [56] permiten medir el impacto que tiene un determinado número de factores de forma individual y conjunta en una o varias variables de respuesta. Cada factor k cuenta con dos posibles estados, denominados nivel ALTO y nivel BAJO; y, por cada combinación de factor y estado se realizan r repeticiones del experimento a fin de aislar errores que pudieran producirse en la ejecución.

La evaluación de la estrategia basada en modelos en BOINC-MGE se realizó usando 2 factores y 3 repeticiones por cada nivel, sumando en total $2^2 3 = 12$ ejecuciones de experimentos diferentes. Se pretende evaluar el impacto que tienen los factores escogidos en 4 variables de respuesta. Los factores y variables de respuesta descritos a continuación fueron seleccionados de tal forma que se pudiera comparar el impacto de usar la estrategia de planificación de BOINC-MGE con respecto a BOINC en la gestión de la batería de los dispositivos y el tiempo total de finalización de procesamiento de una imagen completa.

9.2.1 Factores y niveles

Los factores escogidos son el uso de la extensión BOINC-MGE y la cantidad de batería disponible en los dispositivos.

- **Factor A: Batería**
  Indica si los dispositivos móviles usados tendrán un límite o no en la cantidad de batería disponible. Tener un límite de batería indica que algunos de los dispositivos cuentan con poca batería disponible al momento de registrarse en la grid. Este es un escenario posible en BOINC ya que el usuario puede configurar en la aplicación cuanta batería está dispuesto a compartir. Este factor permitirá evaluar la capacidad de la estrategia para gestionar de forma eficiente este recurso en los dispositivos. Los dos niveles se describen a continuación.
  - *Nivel ALTO (+1):* Los dispositivos no tienen límite de batería, lo cual indica que tienen suficiente carga para finalizar de forma holgada todos los trabajos que se le asignen.

  - *Nivel BAJO (-1):* Algunos de los dispositivos cuentan con poca batería disponible al momento de ingresar a la grid (20% de carga disponible), con lo cual luego de algún tiempo no tendrán suficiente carga para ejecutar nuevos trabajos.
- **Factor B: Uso de BOINC-MGE**
  Factor que permite variar el algoritmo de planificación a usar por parte del proyecto de computación. Pudiendo seleccionar entre el algoritmo de planificación con el que cuenta BOINC o la estrategia SEAS que incorpora la extensión BOINC-MGE desarrollada en el presente trabajo. Los dos niveles de este factor son:
  - *Nivel ALTO (+1)*: La grid usa BOINC-MGE y por lo tanto la estrategia SEAS para la planificación de los trabajos en los dispositivos móviles.
  - *Nivel BAJO (-1)*: La extensión BOINC-MGE se deshabilita en el proyecto de computación, por lo tanto, se usa el algoritmo de planificación de BOINC. Se hace la comparación con el algoritmo de planificación de BOINC debido a que es la estrategia que será reemplazada por la extensión MGE y es la usada por defecto en cualquier proyecto de BOINC sin importar si se trata de una grid móvil o no.

9.2.2 Variables de respuesta

Las variables indicadas a continuación fueron medidas en cada ejecución del experimento para evaluar el impacto que sobre ellas tienen los factores mencionados anteriormente.

- **Cantidad total de trabajos**
  Indica la cantidad de trabajos que en su totalidad fueron generados por el planificador y enviados a los dispositivos móviles. La cantidad total de trabajos aumenta a medida que los dispositivos no pueden finalizarlos y es necesario generar nuevas instancias para entregarlos a otros dispositivos.
- **Porcentaje promedio de batería consumida**
- Es el promedio total de batería que se consume en los dispositivos móviles dentro de la grid, desde que se inicia la ejecución de los trabajos hasta que se finalizan. Es decir, es la cantidad promedio de energía desperdiciada por dispositivo para procesar una imagen de gran tamaño.
- **Tiempo total de procesamiento**
  Indica el tiempo en minutos que se tarda la grid en procesar todos los trabajos generados a partir de una imagen.
- **Cantidad de trabajos reasignados**
  Número de trabajos que luego de ser enviados por el planificador a los dispositivos móviles, no fueron finalizados de forma correcta, debido por ejemplo a la culminación del tiempo límite de procesamiento o al agotamiento de la batería disponible en un dispositivo.

9.2.3 Configuración del experimento

Para la ejecución del experimento se configuró una grid móvil con las siguientes características.

- **Servidor BOINC**
  Máquina Virtual de Google Cloud Computing desplegada en Carolina del Sur (Estados Unidos) con sistema operativo Linux Debian, 1 CPU virtual, 1,7GB de memoria RAM y 20GB de espacio en disco duro.
- **Dispositivos móviles**
  5 dispositivos móviles de gama media fueron usados como nodos de computo, con las siguientes características:
  - 4 celulares Samsung Galaxy S4 con procesador OctaCore a 1.6GHz y batería de 2600mAh.
  - 1 celular Motorola Moto G con procesador QuadCore a 1.2GHz y batería de 2070mAh.
- **Imagen por segmentar**
  Mediante el uso de la aplicación **canny_edge_app** se detectarán los bordes de una imagen de mapa de bits de 296MB de tamaño.
- **Cantidad de trabajos a generar**
  La aplicación **cannyedge_work_generator** dividirá la imagen original en 50 sub-imágenes, correspondiendo cada una a un trabajo que será ejecutado en el conjunto de dispositivos móviles. Cada trabajo tarda en promedio unos 12 minutos en ser ejecutado, sin contar el tiempo de descarga y subida de archivos al servidor.
- **Combinaciones**
  Debido a que se varían siempre 2 factores, la cantidad de combinaciones que se deben probar en cada repetición del experimento son 4. La **Tabla 4** muestra el resumen y descripción de las cuatro combinaciones evaluadas.

Tabla 4. Combinaciones del experimento a ejecutar

Combinación	Factor A	Factor B	Descripción
Con límite de batería y BOINC	BAJO	BAJO	Hay restricciones de batería en algunos dispositivos (2 con 20%) No se usa BOINC-MGE
Con límite de batería y BOINC-MGE	BAJO	ALTO	Hay restricciones de batería en algunos dispositivos (2 con 20%) Se usa BOINC-MGE
Sin límite de batería y BOINC	ALTO	BAJO	No hay restricciones de batería en los dispositivos. No se usa BOINC-MGE
Sin límite de batería y BOINC-MGE	ALTO	ALTO	No hay restricciones de batería en los dispositivos. Se usa BOINC-MGE

9.2.4 Resultados de la ejecución del diseño experimental

A continuación, se presentan los resultados obtenidos luego de la ejecución del diseño experimental para cada una de las variables de respuesta.

- **Cantidad total de trabajos**
  Para el experimento realizado se espera que la cantidad de trabajos generados observada sea igual a 50, que corresponde al total de tareas generadas en el servidor. Si este fuera el caso, indicaría que los dispositivos procesaron de forma eficiente los trabajos sin desperdiciar recursos de cómputo y batería.
  La **Tabla 5** presenta los resultados obtenidos para cada repetición del experimento y cada combinación de los factores.

Tabla 5 Cantidad total de trabajos por cada combinación de factores

Combinación	Cantidad total de trabajos						
	Promedio	Repetición			Error		
		1	2	3	R1	R2	R3
Con límite de batería y BOINC	53,67	53	54	54	-0,67	0,33	0,33
Con límite de batería y BOINC-MGE	50,33	51	50	50	0,67	-0,33	-0,33
Sin límite de batería y BOINC	50	50	50	50	0	0	0
Sin límite de batería y BOINC-MGE	50	50	50	50	0	0	0

Se puede observar que cuando los dispositivos no tienen limitaciones de batería, la cantidad de trabajos generados es el esperado sin importar el planificador que se use. Por el contrario, cuando hay limitaciones de batería en algunos dispositivos, el número de trabajos generados por el planificador de BOINC-MGE se acerca mucho más al valor ideal, esto se debe a que BOINC-MGE monitorea el estado de batería de los dispositivos, con lo cual evita enviar trabajos a dispositivos que no podrán finalizarlos antes de quedarse sin batería disponible. Además, en dos de los 3 experimentos realizados BOINC-MGE genera el número de trabajos esperados mientras que el planificador de BOINC siempre pierde trabajos en los dispositivos por cuenta de la batería, con lo cual deben generarse nuevas instancias de trabajos para ser finalizados en otros dispositivos.

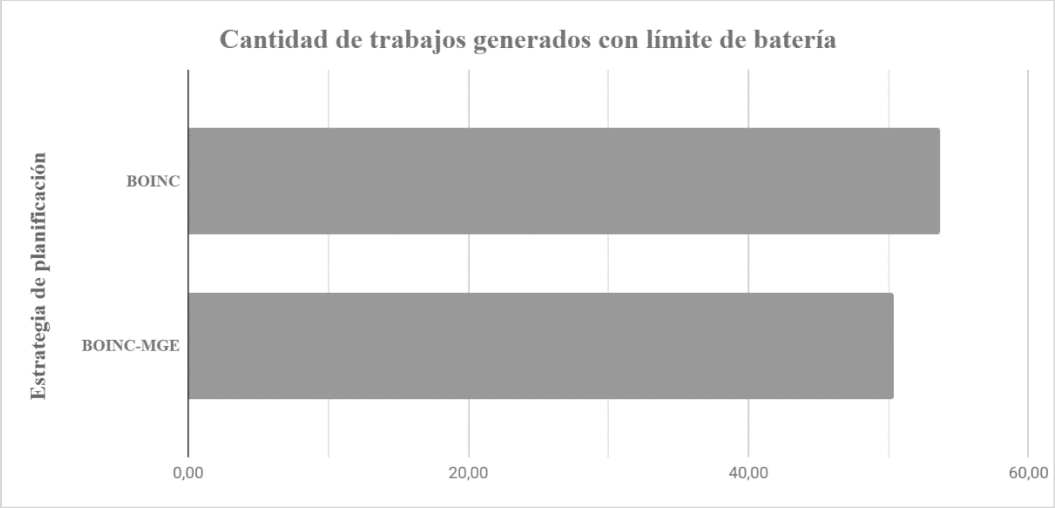


Figura 11 - Cantidad de trabajos generados con límite de batería

En un diseño experimental de tipo factorial $2^k r$ la suma de los cuadrados totales SST se representa como la suma de los cuadrados de cada factor SSA, SSB, su interacción SSAB y la sumatoria de los errores al cuadrado SST.

$$SST = SSA + SSB + SSAB + SSE$$

$$SST = 12 + 8{,}33 + 8{,}33 + 1{,}33 = 30$$

El porcentaje de variación debido a un determinado factor o impacto sobre la variable respuesta se calcula como la relación entre la suma de los cuadrados del factor y la sumatoria total, por ejemplo, el impacto que tiene el factor A sobre la variable respuesta cantidad de trabajos es:

$$qA = 100 * \frac{SSA}{SST} = 100 * \frac{12}{30} = \mathbf{40}\%$$

El impacto que tiene cada uno de los factores y sus interacciones en la variable respuesta se muestra en la **Tabla 6** junto con el porcentaje de error obtenido. No tener límite de batería (factor A) es el que mayor incidencia tiene en la cantidad de trabajos generados, seguido en igual proporción por el uso de BOINC-MGE (factor B) y la interacción entre ambos. Como se mencionó antes, cuando no hay límite de batería los dos planificadores tienen igual rendimiento, sin embargo, en un contexto donde los voluntarios tienen limitaciones de batería el planificador BOINC-MGE consigue reducir la cantidad de trabajos generados por la grid para finalizar el procesamiento de la imagen.

Tabla 6. Impacto de factores en la variable respuesta cantidad total de trabajos

Variable: Cantidad total de trabajos	
Factor	**Impacto**
qA (Batería)	**40,0%**
qB (BOINC-MGE)	27,78%
qAB	27,78%
Error	4,44%

- **Porcentaje promedio de batería consumida**
  Esta variable de respuesta mide la cantidad promedio de batería (medida en porcentaje disponible) que consumieron los dispositivos móviles durante la ejecución de los trabajos. El total de batería consumida es la diferencia entre la batería disponible al momento de asignación del primer trabajo y la batería disponible al momento de finalizar el último trabajo enviado al servidor. Debido a que todos los dispositivos eran de características similares, se tomó el promedio de consumo entre los 5 teléfonos usados.
  La **Tabla 7** muestra las mediciones promedio obtenidas para cada repetición y combinación de factores.

Tabla 7 Consumo promedio de batería por cada combinación de factores

Combinación	% de Batería promedio consumida						
	Promedio	Repetición			Error		
		1	2	3	R1	R2	R3
Con límite de batería y BOINC	25	24	25	26	-1	0	1
Con límite de batería y BOINC-MGE	20	19	21	20	-1	1	0
Sin límite de batería y BOINC	26	27	26	25	1	0	-1
Sin límite de batería y BOINC-MGE	18,67	19	19	18	0,33	0,33	-0,67

Tanto si hay límite de batería o no, BOINC-MGE consume en promedio menor cantidad de batería en los dispositivos. Esto se puede explicar debido a que el algoritmo de planificación de BOINC-MGE monitorea el estado de los dispositivos y evita enviar trabajos que no podrán ser terminados a tiempo, evitando de esta forma que se gasten recursos de CPU y batería en un dispositivo que no alcanzará a finalizar un trabajo. Además, usando este mismo esquema, si un dispositivo tiene suficiente batería disponible para terminar varios trabajos, BOINC-MGE los asigna de forma anticipada, de tal forma que al finalizar un trabajo ya se cuente con los

archivos y datos necesarios para iniciar el procesamiento del siguiente asignado, generando de esta forma que se malgaste menos batería por tiempo ocioso del dispositivo. Las Figuras 12 y 13 permiten apreciar mejor el impacto positivo que tiene el uso de BOINC-MGE en el consumo promedio de batería de los dispositivos.

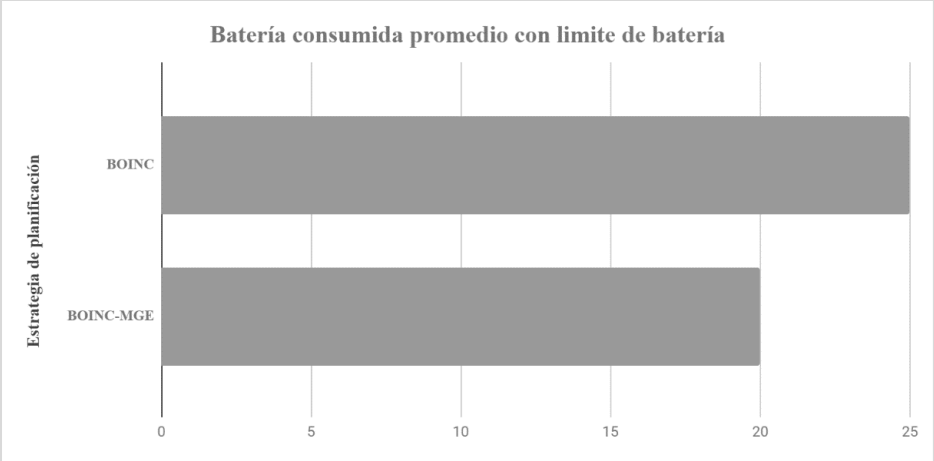


Figura 12 - Batería promedio consumida con límite de batería

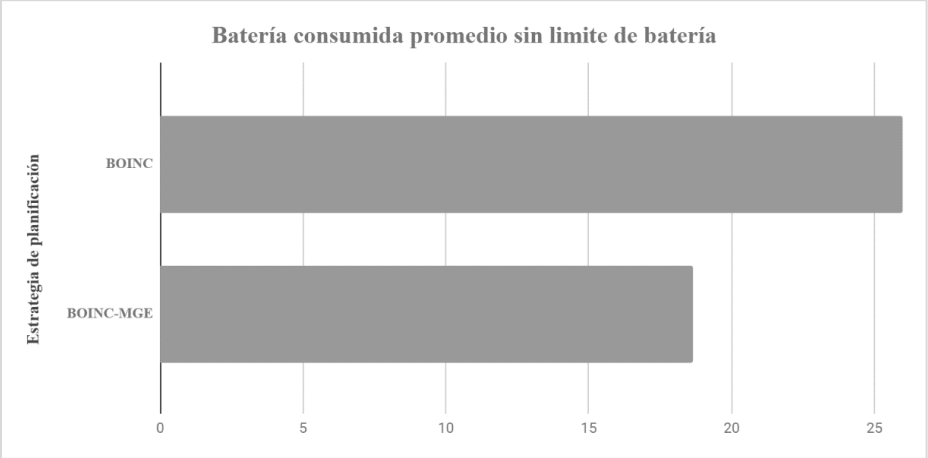


Figura 13 - Batería promedio consumida sin límite de batería

Tabla 8. Impacto de factores en el consumo de batería

Variable: % de batería consumida	
Factor	**Impacto**
qA (Batería ilimitada)	0,07%
qB (BOINC-MGE)	**91,33%**
qAB	3,27%
Error	5,34%

La **Tabla 8** muestra el porcentaje de impacto de cada factor junto con el porcentaje de error obtenido. Como es de esperarse de acuerdo con los resultados resumidos anteriormente, el único factor que genera un impacto significativo en el consumo de batería es el uso del planificador BOINC-MGE.

- **Tiempo total de procesamiento**
  En esta variable se mide el tiempo total que tarda la grid en terminar de procesar todos los trabajos que conforman una imagen. Para ello se toma la diferencia de tiempo (en minutos) desde que se envió el primer trabajo a un dispositivo hasta que se recibió el ultimo resultado exitoso. La **Tabla 9** muestra los resultados obtenidos.

Tabla 9 Tiempo total de procesamiento por cada combinación de factores

Combinación	Tiempo total de procesamiento (minutos)						
	Promedio	Repetición			Error		
		1	2	3	R1	R2	R3
Con límite de batería y BOINC	97,56	98,68	106,68	87,30	1,13	9,13	-10,26
Con límite de batería y BOINC-MGE	66,69	64,92	70,17	64,98	-1,77	3,48	-1,71
Sin límite de batería y BOINC	77,59	81,10	75	76,67	3,51	-2,59	-0,92
Sin límite de batería y BOINC-MGE	52,58	53,28	51,47	53,00	0,70	-1,12	0,42

En un contexto de batería limitada el tiempo total de finalización de los 50 trabajos generados es mayor a su contraparte en la cual no existe limitante de batería, tanto si se usa BOINC como si se usa BOINC-MGE. Sin embargo, en igualdad de condiciones de batería (limitada o no) BOINC-MGE reduce el tiempo total de procesamiento alrededor de un 30% con respecto a BOINC. Para el caso puntual de procesamiento de 50 trabajos esto corresponde a una reducción de tiempo de hasta 30 minutos, lo cual se debe a que BOINC-MGE al no despachar trabajos a dispositivos que no podrán finalizarlos no se generan tiempos muertos de espera por parte del servidor para reasignar dichos trabajos a nuevos dispositivos, BOINC solo reasigna trabajos cuando recibe un error de ejecución por parte de un dispositivo o cuando se vence el tiempo límite de espera, el cual en este experimento se encuentra configurado en 30 minutos. Las figuras 14 y 15 permiten apreciar el mejor rendimiento que aporta BOINC-MGE sobre BOINC en las dos condiciones de batería probadas.

La **Tabla 10** muestra el porcentaje de impacto de cada factor, la cual permite observar que el uso o no de BOINC-MGE es el factor que mayor efecto produce sobre el consumo de batería.

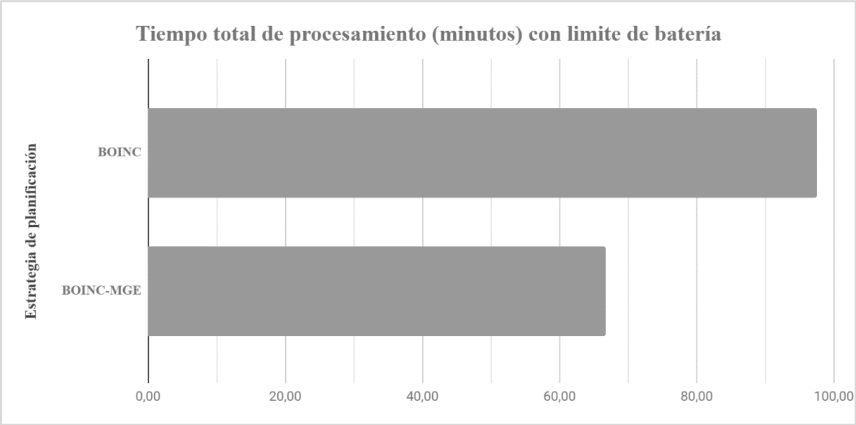


Figura 14 - Tiempo total de procesamiento con límite de batería

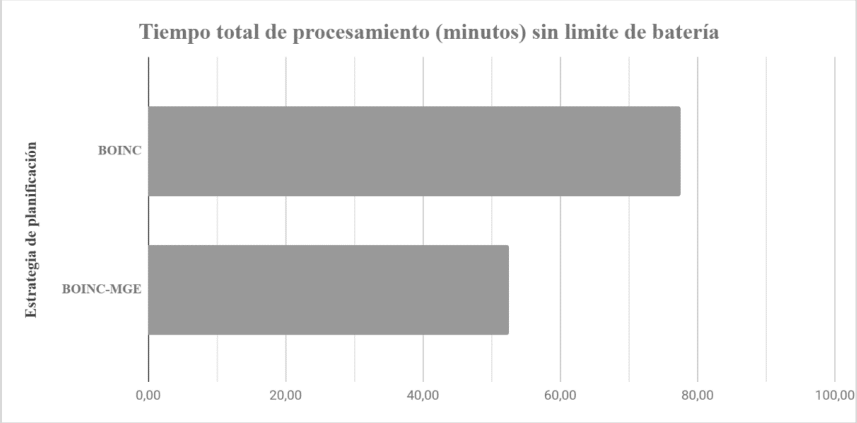


Figura 15 - Tiempo total de procesamiento sin límite de batería

Tabla 10. Impacto de factores en el tiempo de procesamiento

Variable: Tiempo total de procesamiento	
Factor	**Impacto**
qA (Batería ilimitada)	25,11%
qB (BOINC-MGE)	**67,52%**
qAB	0,74%
Error	6,62%

- **Cantidad de trabajos reasignados**
  Esta variable corresponde a la cantidad de trabajos que tuvieron que ser reasignados a otros dispositivos debido a su no finalización. Los resultados como era de esperarse son proporcionales a la cantidad de trabajos generados medidos anteriormente. En el caso en el que no hay límite de batería, culminan todos los trabajos donde fueron asignados y no es necesaria la replanificación; en un contexto de batería limitada BOINC-MGE se comporta mejor con respecto a BOINC. El resumen de las mediciones obtenidas y el impacto de cada factor en la variable de respuesta se muestran en las Tablas 11 y 12 respectivamente.

Tabla 11 Trabajos reasignados por cada combinación de factores

Combinación	Cantidad de trabajos reasignados						
	Promedio	Repetición			Error		
		1	2	3	R1	R2	R3
Con límite de batería y BOINC	4	4	4	4	0	0	0
Con límite de batería y BOINC-MGE	0,33	1	0	0	0,67	-0,33	-0,33
Sin límite de batería y BOINC	0	0	0	0	0	0	0
Sin límite de batería y BOINC-MGE	0	0	0	0	0	0	0

Tabla 12. Impacto de factores en la cantidad de trabajos reasignados

Variable: Cantidad de trabajos reasignados	
Factor	**Impacto**
qA (Batería ilimitada)	40,33%
qB (BOINC-MGE)	**28,88%**
qAB	28,88%
Error	1,91%

9.3 Evaluación de estrategia RL SEAS REPL basada en réplicas

Para la evaluación de la estrategia basada en réplicas se hizo uso igualmente de un diseño experimental factorial con 2 factores, los cuales en este caso tienen diferente cantidad de niveles. Se trata entonces de un diseño experimental de tipo full factorial de dos factores, el primer factor *k* (escenario) consta de 4 niveles y el segundo factor *j* (estrategia de réplicas) consta igualmente de 4 niveles, sin usar en este caso repeticiones por cada combinación, con lo cual el total de

experimentos a realizar es $4^k * 4^j = 4^1 * 4^1 = 16$. Los factores y niveles se describen a continuación.

9.3.1 Factores y niveles

Los factores escogidos son el escenario de ejecución y la estrategia de generación de réplicas.

- **Factor A: Escenario**
  Indica el escenario en el que se ejecutan los trabajos, el cual puede ser un escenario ideal donde no hay desconexiones y otros tres escenarios que modelan igual número de condiciones de desconexiones en los dispositivos participantes en la grid. Los cuatro niveles se describen a continuación.
  - *Nivel 1: Sin desconexiones.* Representa un escenario ideal en el cual los dispositivos no se desconectan durante todo el tiempo que tardan en terminar todos los trabajos de la aplicación.
  - *Nivel 2: Muy inestable.* Escenario en el cual 2 de los 5 dispositivos disponibles se desconectan de forma aleatoria cada 10 minutos, tiempo en el cual no se alcanza a finalizar un trabajo enviado a un dispositivo (este tiempo es de aproximadamente 12 minutos).
  - *Nivel 3: Estabilidad media*. Corresponde a un escenario donde se realizan desconexiones aleatorias de dos dispositivos móviles cada 15 minutos. Este tiempo es suficiente para que se termine al menos un trabajo antes de producirse una desconexión por parte de un dispositivo y menor al tiempo de vencimiento establecido en el servidor de BOINC para las tareas enviadas a los dispositivos.
  - *Nivel 4*: *Pocas desconexiones.* Escenario en el cual se realizan desconexiones de dos dispositivos seleccionados de forma aleatoria cada 30 minutos, tiempo suficiente para que se finalicen al menos dos trabajos en un dispositivo antes de desconectarse, pero lo suficientemente alto para que los trabajos enviados a un dispositivo que luego se desconecta terminen siendo marcados por el servidor como no finalizados debido al vencimiento del tiempo límite de respuesta, configurado en 30 minutos.
- **Factor B: Estrategia de réplicas**
  Con este factor se varía la estrategia de réplicas usada en el planificador de trabajos para determinar cuántas instancias adicionales se generan por cada una de las 50 tareas que se ejecutan en el experimento. Se pretende evaluar tres configuraciones diferentes que soporta BOINC y la estrategia de planificación basada en replicas RL REPL propuesta en el presente trabajo de investigación. Se debe tener en cuenta que, aunque las réplicas de BOINC no fueron implementadas para implementar tolerancia a fallas por desconexiones, sí es posible usarlas con este propósito en cualquier proyecto, razón por la cual se hace la comparación de BOINC-MGE con las correspondientes alternativas de BOINC. Los cuatro niveles se describen a continuación.
  - *Nivel 1: BOINC con una réplica.* Representa la configuración por defecto de BOINC en la cual por cada trabajo enviado desde el servidor a un

dispositivo no se generan replicas adicionales a ser ejecutadas por otros dispositivos de forma simultánea. Sin embargo, si se generan reintentos sobre la misma replica en caso de no finalización en el tiempo límite establecido de 30 minutos. Se considera en este caso que "una réplica" es el trabajo original enviado al dispositivo.

- *Nivel 2: BOINC con dos réplicas adaptativas.* Modelo de réplicas de BOINC en el cual se genera una réplica adicional por cada trabajo enviado desde el servidor a un dispositivo que es considerado no confiable por el algoritmo de planificación de BOINC. La forma como se cataloga a un dispositivo como no confiable se describe en la sección **4.2.2 Réplicas en BOINC.**
- *Nivel 3: BOINC con tres réplicas adaptativas.* Similar al escenario anterior, pero en este caso BOINC genera dos replicas adicionales a la unidad de trabajo original siempre que considera que un dispositivo no es confiable.
- *Nivel 4: BOINC-MGE con 1 a 3 réplicas dinámicas.* Escenario de réplicas de la estrategia RL REPL. Tal como se mencionó en la sección **8.2.1 Descripción algoritmo estrategia RL REPL**, en RL REPL la cantidad de réplicas se calcula de forma dinámica mediante un algoritmo de aprendizaje reforzado que tiene en cuenta las desconexiones de los dispositivos y el consumo de batería debido a las réplicas generadas anteriormente. Cuando el servidor envía una unidad de trabajo por primera vez determina el número de réplicas a generar, el cual puede ser 1 (no se generan replicas adicionales), 2 (se genera una réplica adicional que será entregada a otro dispositivo) o 3 (se generan dos replicas adicionales que serán entregadas a dos dispositivos diferentes).

9.3.2 Variables de respuesta

Las variables de respuesta indicadas a continuación se tomaron en cada ejecución del experimento y al final fueron consolidadas para su evaluación.

- **Porcentaje promedio de batería consumida**
  Es el promedio total de batería que se consume en los dispositivos móviles dentro de la grid, desde que se inicia la ejecución de los trabajos hasta que se finalizan. Es decir, es la cantidad promedio de batería desperdiciada por dispositivo para procesar una imagen de gran tamaño.
- **Tiempo total de procesamiento**
  Indica el tiempo en minutos que se tarda la grid en procesar todos los trabajos generados a partir de una imagen.

9.3.3 Configuración del experimento

Para la ejecución del experimento se configuró una grid móvil con las siguientes características.

- **Servidor BOINC**
  Máquina Virtual de Google Cloud Computing desplegada en Carolina del Sur (Estados Unidos) con sistema operativo Linux Debian, 1 CPU virtual, 1,7GB de memoria RAM y 20GB de espacio en disco.
- **Dispositivos móviles**
  5 dispositivos móviles de gama media fueron usados como nodos de computo cuyas características son las siguientes:
  - 4 celulares Samsung Galaxy S4 con procesador OctaCore a 1.6GHz y batería de 2600mAh.
  - 1 celular Motorola Moto G con procesador QuadCore a 1.2GHz y batería de 2070mAh.
  - Todos con arquitectura de CPU ARM.
  - En cada dispositivo se usaron 2 CPUs para el computo de los trabajos, con lo cual se ejecutarán en simultaneo un máximo de dos trabajos en cada teléfono.
- **Imagen por segmentar**
  Haciendo uso del *benchmark* descrito en la sección **9.1.2 Benchmark para el procesamiento de la imagen** se procesará una imagen de mapa de bits de 296MB de tamaño.
- **Cantidad de trabajos a generar**
  La aplicación **cannyedge_work_generator** dividirá la imagen original en 50 sub-imágenes, correspondiendo cada una a un trabajo que será ejecutado en el conjunto de dispositivos móviles. Cada trabajo tarda en promedio unos 12 minutos en ser ejecutado, sin contar el tiempo de descarga y subida de archivos al servidor.
- **Desconexiones**
  En los escenarios en los que hay desconexiones se inicia la ejecución de los trabajos con todos los dispositivos conectados, luego cada cierto tiempo (10, 15 o 30 minutos) se seleccionan dos dispositivos al azar y se apagan para evitar que sigan generando comunicación con el servidor y procesando trabajos. Al transcurrir nuevamente el tiempo determinado se vuelven a seleccionar dos dispositivos al azar para ser desconectados y se integran los otros dos que fueron apagados en el ciclo anterior. Se repite el mismo proceso hasta que en el servidor se encuentren todos los 50 trabajos finalizados. Es importante aclarar que debido a que no se implementó en la aplicación de prueba el uso de puntos de control locales, cuando un dispositivo se reconecta y tiene trabajos que no fueron finalizados, los mismos serán ejecutados nuevamente en su totalidad.

9.3.4 Resultados de ejecución

A continuación, se describen los resultados obtenidos para cada una de las variables de respuesta medidas, describiendo los valores obtenidos y el efecto causado en cada una de ellas por los factores seleccionados en sus diferentes niveles.

- **Tiempo total de procesamiento**
  La **Tabla 13** resume el valor total de procesamiento de los 50 trabajos medido en minutos para cada una de las combinaciones del diseño experimental, las columnas

corresponden a los escenarios de desconexión y las filas a las estrategias para el cálculo de réplicas usado. Además, se agregan dos columnas y filas adicionales que muestran el promedio total de procesamiento obtenido por cada nivel de los factores y el efecto que tiene cada uno con respecto al promedio global. Finalmente, la celda donde se interceptan los promedios de cada factor muestra el tiempo total de ejecución promedio para todas las combinaciones de factores, el cual es de **191,23** minutos.

Tabla 13 Tiempo total de procesamiento en diferentes escenarios

Tiempo total de procesamiento (minutos)						
Combinación	**Sin desconexiones**	**Muy inestable**	**Estabilidad media**	**Pocas desconexiones**	***Promedio***	***Efecto***
BOINC – 1 REPLICA	64,43	182,92	138,47	103,63	*122,36*	*-68,86*
BOINC – 2 REPLCIAS	99,38	322,78	260,75	208,03	*222,74*	*31,51*
BOINC – 3 REPLICAS	189,73	402,07	371,63	302,93	*316,59*	*125,36*
BOINC-MGE – 1 a 3 REPLICAS	63,15	132,88	122,97	93,87	*103,22*	*-88,01*
Promedio	*104,18*	*260,16*	*223,45*	*177,12*	***191,23***	
Efecto	*-87,05*	*68,94*	*32,23*	*-14,11*		

Se puede observar que en promedio los menores tiempos de ejecución se obtienen cuando se usa la estrategia de réplicas propuesta en el presente trabajo, es decir BOINC-MGE con el número de réplicas calculas de forma dinámica entre 1 y 3. El tiempo promedio obtenido con BOINC-MGE es de 103,22 minutos, es decir 88,01 minutos (1 hora y 20 minutos) más rápido que el promedio general. En términos de escenarios de desconexión, se observa como era de esperarse que en promedio se obtienen menores tiempos de ejecución cuando no hay desconexiones de ninguno de los dispositivos (104,18 minutos) seguido del escenario de pocas desconexiones (177,12), en el cual los dispositivos tienen más tiempo disponible para la ejecución de los trabajos (30 minutos) antes de desconectarse.

El efecto de cada nivel por factor indica la cantidad de minutos en promedio que se adicionan o disminuyen sobre el promedio total cada vez que se ejecutan los trabajos baso la condición medida, y se calcula de la siguiente forma:

$$\alpha_j = \overline{y_{.j}} - \bar{y}$$
$$\beta_i = \overline{y_{.i}} - \bar{y}$$

α_j es el efecto del factor A en el nivel j y es la diferencia entre el promedio del factor en dicho nivel $y_{.j}$ y el promedio total $\bar{y}$. De forma similar, β_i es el efecto del factor B en el nivel i.
En el experimento ejecutado, se puede ver por medio de los efectos calculados, que ejecutar los trabajos en un escenario sin desconexiones genera un efecto positivo de 87,05 minutos menos que el promedio general (o lo que es lo mismo, se tardan 87,05 minutos menos en terminar todos los trabajos en esta condición). Por otro lado, al usar la estrategia de generación de réplicas de BOINC con tres replicas adaptativas genera que en promedio se necesiten 125,36 minutos (2 horas y 25 minutos) más que el promedio general para finalizar los trabajos. Se puede observar que la estrategia RL REPL es, en promedio, mejor que todas las posibles combinaciones de réplicas de BOINC; esto se debe a que la estrategia puede detectar mediante su algoritmo los diferentes estados de desconexiones de los dispositivos y generar la cantidad de réplicas adecuadas que garantizan un menor tiempo de finalización de todos los trabajos.

En la **Figura 16** se puede observar de manera gráfica el tiempo total de procesamiento que se obtiene por cada estrategia de réplicas en los diferentes escenarios de desconexiones. Las estrategias de BOINC que generan más de una réplica tienen significativamente mayores tiempos de procesamiento en cualquiera de los escenarios. La estrategia de BOINC-MGE, por otro lado, es la que genera menores tiempos de procesamiento incluso en escenarios de desconexiones altas (Muy inestable y Estabilidad media) con respecto a la estrategia de BOINC que no genera réplicas adicionales. La línea punteada en el centro marca el valor promedio obtenido para todas las combinaciones posibles, el cual sirve como referencia para evaluar los tiempos obtenidos por las diferentes estrategias con respecto a dicho valor.

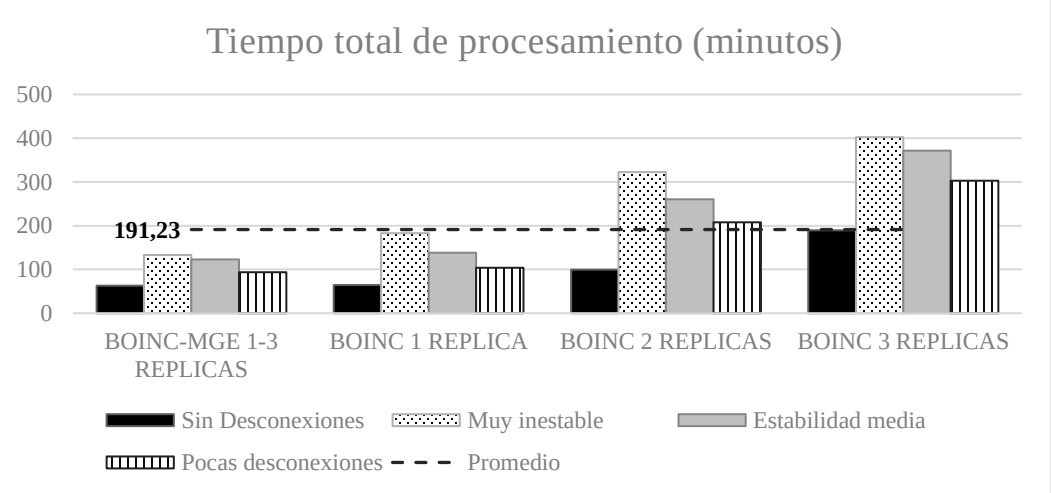


Figura 16 Tiempo total de procesamiento en diferentes escenarios

El porcentaje de variación sobre la variable de respuesta debido a cada factor se calcula usando la suma de los cuadrados. SSY es la suma de los cuadrados de todas las 16 mediciones correspondientes a los diferentes escenarios, SS0 es el producto de la cantidad de niveles de cada factor por el cuadrado del promedio general, SSA y SSB es el producto de la cantidad de niveles del factor A o B por la suma de los cuadrados de los efectos obtenidos en cada uno de los niveles de cada factor. En las siguientes ecuaciones, a y b corresponden al número de niveles de los factores A y B respectivamente, los cuales son en ambos casos 4.

$$SSY = \sum_{ij} y_{ij}^2 = 765971{,}54$$

$$SS0 = abμ^2 = 4 * 4 * 192{,}23^2 = 585084{,}76$$

$$SSA = b\sum_{j} \alpha_j^2$$

$$SSA = 4 * [(-87{,}05)^2 + (-68{,}94)^2 + (-32{,}23)^2 + (-14{,}11)^2] = 54271{,}38$$

$$SSB = a\sum_{i} \beta_i^2 = 116789{,}40$$

Con los cálculos anteriores se puede obtener la variación total SST y la variación debida a errores de medición SSE de la siguiente forma.

$$SST = SSY - SS0 = 765871{,}54 - 585084{,}76 = 180886{,}78$$

$$SSE = SST - SSA - SSB = 180886{,}78 - 54271{,}38 - 116789{,}40 = 9826$$

Finalmente podemos obtener el porcentaje de variación explicado por el factor A, escenario de desconexiones, el cual es:

$$100 * \frac{SSA}{SST} = 100 * \frac{54271{,}38}{180886{,}78} = \mathbf{30}\%$$

De forma similar el porcentaje de variación explicado por el factor B, estrategia de planificación basada en replicas es:

$$100 * \frac{SSB}{SST} = 100 * \frac{116789{,}40}{180886{,}78} = \mathbf{64{,}56}\%$$

Finalmente, el porcentaje de variación no explicable es:

$$100 * \frac{SSE}{SST} = 100 * \frac{9826{,}0}{180886{,}78} = \mathbf{5{,}43}\%$$

Al ver los porcentajes de variación, podemos concluir que la selección de la estrategia de planificación para el cálculo de réplicas es un factor importante que afecta el tiempo total de procesamiento de los trabajos de forma considerable, el segundo factor que afecta esta variable respuesta es el escenario de desconexiones en el cual se ejecute. Un porcentaje de variación no explicable de solo el 5,43% nos indica que hicimos una selección apropiada de los factores para la realización del experimento, debido a que cerca del 95% de las variaciones en la variable de respuesta son explicables por los factores A y B.

- **Porcentaje promedio de batería consumida**
  La intención de medir esta variable respuesta es verificar el impacto que produce sobre el consumo promedio de batería en todos los dispositivos la generación de réplicas para disminuir el tiempo de finalización de los trabajos. La **Tabla 14** muestra los resultados obtenidos, así como los efectos por cada nivel de los factores en la variable de respuesta.

Tabla 14 Consumo promedio de batería en diferentes escenarios

Porcentaje promedio de batería consumida						
Combinación	**Sin desconexiones**	**Muy inestable**	**Estabilidad media**	**Pocas desconexiones**	***Promedio***	***Efecto***
BOINC – 1 REPLICA	27	41	31	24	*30,75*	*-13*
BOINC – 2 REPLCIAS	36	56	65	48	*51,25*	*7,50*
BOINC – 3 REPLICAS	53	74	76	62	*66,25*	*22,50*
BOINC-MGE – 1 a 3 REPLICAS	22	31	30	24	*26,75*	*-17*
Promedio	*34,50*	*50,50*	*50,50*	*39,50*	***43,75***	
Efecto	*-9,25*	*6,75*	*6,75*	*-4,25*		

Se puede observar que el promedio general de consumo de batería sin importar la estrategia de planificación de réplicas seleccionada ni el escenario de desconexiones en el cual se ejecute es de 43,75% por dispositivo móvil. Este valor se reduce 17% si se usa la estrategia de planificación propuesta en el presente trabajo de investigación, el cual es mucho mejor que cualquiera de los escenarios de BOINC. En cuanto al escenario de desconexión, como era de esperarse se obtienen menores porcentajes de consumo cuando no hay desconexiones en la grid o cuando las desconexiones se presentan de forma menos frecuente. Al haber escenarios de desconexiones más frecuentes aumenta el consumo de batería

promedio, esto es debido a que más trabajos quedan sin finalizarse y, por lo tanto, deben ser reasignados por el planificador a otros dispositivos, lo que genera un mayor gasto de batería general en la grid.

En la **Figura 17** se muestran gráficamente los promedios de consumo por dispositivo en cada escenario junto con el promedio general marcado en la línea punteada horizontal, la cual permite observar la conveniencia de usar la estrategia propuesta en BOINC-MGE con respecto a cualquiera de las otras opciones evaluadas de BOINC.

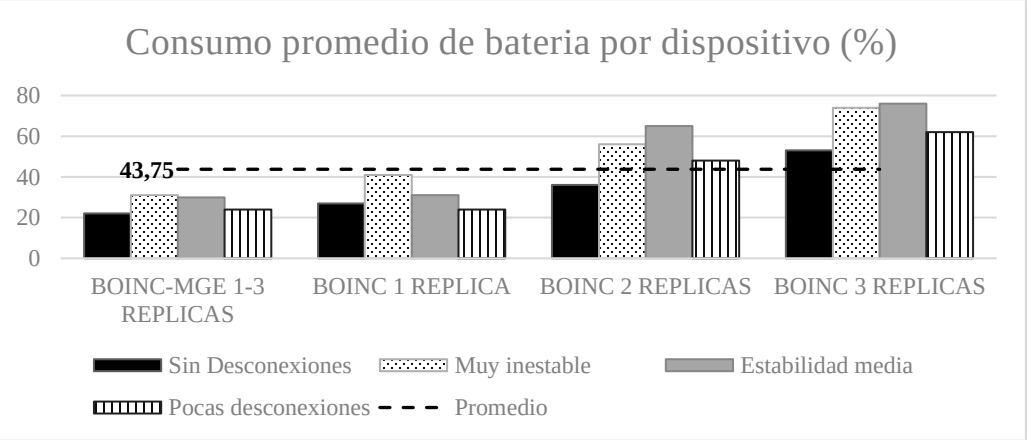


Figura 17 - Consumo promedio de batería por dispositivo

De forma similar a lo realizado para la variable de respuesta anterior, se calculan la suma de los cuadrados de cada factor para obtener el porcentaje de variación sobre el consumo promedio de energía por dispositivo y con ellos se obtienen los porcentajes de variación.

El porcentaje de variación sobre el consumo de energía promedio por dispositivo explicado por el escenario de desconexión en el cual se ejecuten los trabajos es:

$$100 * \frac{SSA}{SST} = 100 * \frac{779}{5109} = \mathbf{15,25}\%$$

El porcentaje de variación explicado por la estrategia de planificación basada en replicas es:

$$100 * \frac{SSB}{SST} = 100 * \frac{4082}{5109} = \mathbf{79,90}\%$$

Este porcentaje se debe en parte a que la estrategia de planificación de BOINC-MGE basada en replicas actúa de forma conjunta con la estrategia basada en modelos descrita en la sección **8.1.1 Descripción algoritmo estrategia SEAS**. En ambas estrategias se usa el estado de la batería de los dispositivos para realizar la planificación de los trabajos y el cálculo del número de réplicas (en el caso de RL

REPL), buscando no solo la finalización de los trabajos en el menor tiempo posible sino también la disminución del consumo de batería general dentro de la grid móvil.

Finalmente, el porcentaje de variación sin explicación es:

$$100 * \frac{SSE}{SST} = \ 100 * \frac{248}{5109} = \mathbf{4,83}\%$$

Se trata de un porcentaje muy pequeño que nos indica igualmente que la selección de los factores a variar durante el experimento fueron los adecuados y explican el valor de la variable respuesta.

9.4 Evaluación del impacto en el servidor debido a BOINC-MGE

En esta sección se evalúa mediante mediciones tomadas en el servidor, el impacto causado al habilitar la extensión MGE y usar las estrategias de planificación basadas en modelos y réplicas en el consumo de memoria y CPU, con respecto a las estrategias de planificación de BOINC. Debido a que los cálculos realizados por BOINC-MGE en el planificador son iguales sin importar el escenario de desconexión y/o el estado de la batería de los dispositivos, se realizan las mediciones en un escenario donde no ocurren desconexiones y en el cual todos los dispositivos tienen suficiente batería para finalizar los trabajos.

A continuación, se describen el escenario de prueba y las variables de respuesta medidas en el servidor, así como las herramientas usadas para su monitoreo.

9.4.1 Factores y niveles

El único factor escogido es la estrategia de planificación usada para asignar los trabajos a los dispositivos móviles.

- **Factor A: Estrategia de planificación**
  Indica la estrategia de planificación usada en el servidor para la asignación de los trabajos a los dispositivos móviles que soliciten tareas a BOINC.
  - *Nivel 1: BOINC.* Estrategia de planificación por defecto de BOINC, ninguna de las características de BOINC-MGE se encuentran habilitadas en este escenario.
  - *Nivel 2: BOINC-MGE*. Estrategia de planificación de BOINC-MGE que incluye tanto la estrategia basada en modelos como la estrategia de cálculo de réplicas.

9.4.2 Variables de respuesta

Las variables de respuesta indicadas a continuación se tomaron durante la ejecución de un conjunto de trabajos para cada uno de los niveles del factor “Estrategia de planificación” indicado anteriormente.

- **Memoria usada**
  Indica la cantidad de memoria RAM usada en el servidor durante todo el periodo de tiempo que se estuvieron ejecutando los trabajos en los dispositivos. Con esta

variable se pretende medir si la extensión MGE genera un mayor consumo de recursos de memoria RAM en el servidor mientras se encuentra realizando los procesos de planificación de trabajos.

- **Uso de CPU**
  Porcentaje de CPU usada en el servidor por los procesos que se ejecutan a nivel de usuario. El planificador de trabajos de BOINC se encuentra en este tipo de procesos, por lo cual medir esta variable permitirá evaluar si el planificador de BOINC-MGE genera un mayor consumo de CPU con respecto a BOINC.

9.4.3 Configuración del experimento

Para la ejecución del experimento se hizo la configuración de una grid móvil con iguales características de trabajos y servidor a las usadas en los experimentos anteriores de evaluación de las estrategias de planificación.

Las mediciones en el servidor fueron tomadas haciendo uso de la herramienta **sysstat** disponible en Linux Debian, la cual toma de forma automática las estadísticas de uso de CPU y memoria, entre otros factores.

9.4.4 Resultados de ejecución

A continuación, se describen los resultados obtenidos para cada una de las variables de respuesta medidas.

- **Memoria usada**
  La **Figura 18** muestra la cantidad total de memoria RAM usada en el servidor durante el tiempo que se estuvieron ejecutando los 50 trabajos en los dispositivos móviles, medida cada 5 minutos. Se puede observar que las dos estrategias presentan el mismo perfil descendente en cuanto al uso de memoria, esto se explica porque a medida que se van finalizando los trabajos en los dispositivos, el número de estructuras que se cargan en memoria del lado del servidor y que mantienen la información de las tareas pendientes de finalización, van disminuyendo.
  La estrategia de BOINC-MGE usa alrededor de un 1% más de memoria que BOINC, debido a que en el caso de la estrategia basada en replicas debe consultar en la base de datos la información de las réplicas anteriores y en el caso de la estrategia basada en modelos se debe consultar la información adicional del dispositivo que permite consultar y guardar las tasas de consumo de batería calculadas por la estrategia SEAS. Este mayor consumo, sin embargo, no es significativo, ya que se trata en promedio de alrededor de unos 20MB más con respecto a BOINC, un valor despreciable si se tiene en cuenta que los servidores actuales cuentan con varios GB de memoria disponible.
- **Uso de CPU**
  El uso de CPU medido en el servidor durante la ejecución de los trabajos para cada una de las estrategias de planificación es mostrado en la **Figura 19**, en ella se puede observar que el consumo general siempre se encuentra por debajo del 0,5% en el caso de BOINC-MGE y BOINC. Se observa un pico de 2,3% en la estrategia de BOINC al inicio del ciclo de planificación, el cual sin embargo es muy bajo e indica

que no hubo, en ninguno de los dos casos, un uso excesivo de la CPU que pudiese degradar el rendimiento en el servidor.

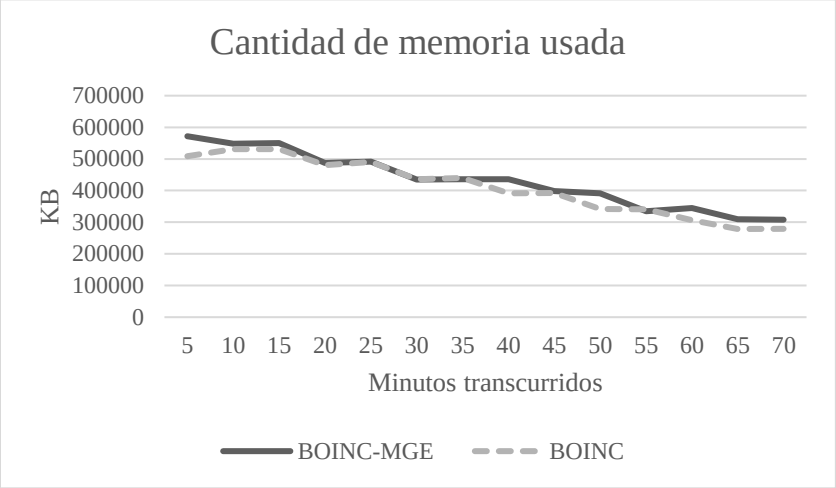


Figura 18 Memoria usada en el servidor

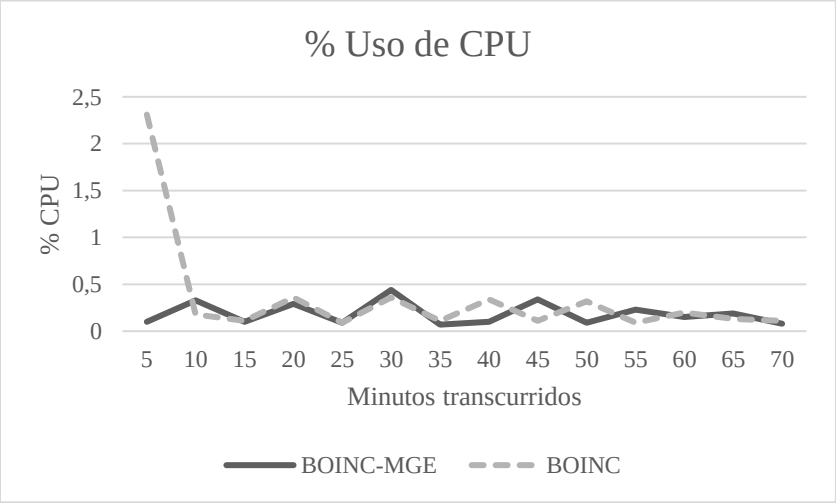


Figura 19 Uso de CPU en el servidor

10 Conclusiones

Los desafíos que surgen cuando se intenta integrar dispositivos móviles a una grid sirvieron de motivación para el presente trabajo de investigación. Al inicio del trabajo se pudo evidenciar cómo la plataforma BOINC a pesar de permitir el uso de dispositivos móviles como plataformas de computo, no soluciona de forma directa muchos de los desafíos identificados. Entre estos se encuentran la gestión de la batería en los dispositivos y la garantía de finalización de los trabajos en escenarios de desconexión.

Como problema de investigación se planteó el diseño y desarrollo de una extensión sobre la plataforma BOINC, llamada BOINC-MGE (Mobile Grid Extension) para dar solución a estos desafíos. Para cumplir dicho propósito, se plantearon cuatro objetivos específicos que pretendían en su conjunto el cumplimiento del objetivo general: La integración en BOINC de estrategias de planificación que considerasen principalmente las desconexiones de los usuarios y el consumo de batería en los dispositivos.

El primer objetivo específico consistía en la evaluación y selección de las dos estrategias de planificación a ser integradas en BOINC. Estudiando estrategias de planificación preventivas, se escogieron de forma preliminar dos categorías para la evaluación: estrategias basadas en modelos que puedan tomar en cuenta el consumo de batería de los dispositivos y estrategias basadas en replicas para lograr culminar los trabajos en escenarios de desconexiones. En la primera categoría se seleccionó la estrategia SEAS, un algoritmo simple y relativamente fácil de implementar que haciendo uso de una estimación de descarga promedio de batería de los dispositivos realiza la asignación de los trabajos, y en la segunda categoría se usó una estrategia basada en técnicas de aprendizaje reforzado para el cálculo del número de réplicas a generar que fue adaptada para ser implementada en el modelo de BOINC. Las dos estrategias resultaron seleccionadas utilizando la metodología DAR.

La tarea del segundo objetivo específico consistió en el diseño e implementación de una API de desarrollo cuyo propósito es facilitar la posterior implementación de nuevas estrategias de planificación en BOINC, sin que sea necesario apropiar muchos conceptos internos de la plataforma.

En el desarrollo del tercer objetivo específico se realizó la implementación de las estrategias de planificación haciendo uso de la API de desarrollo y su validación funcional mediante una metodología de desarrollo ágil que permitía ir corrigiendo y mejorando las funcionalidades de BOINC-MGE en ciclos iterativos. Las modificaciones realizadas en BOINC para integrar la API y las estrategias de planificación se cumplieron de forma satisfactoria como se describió en los capítulos **6. Arquitectura y componentes BOINC-MGE**, **7. API para integración de algoritmos de planificación en BOINC-MGE** y **8. Estrategias de planificación en BOINC-MGE**.

Finalmente, para dar cumplimiento al último objetivo del trabajo de investigación se diseñó un protocolo experimental para la evaluación de las dos estrategias de planificación implementadas haciendo uso de un proyecto de prueba de BOINC y dispositivos móviles reales. Los resultados obtenidos muestran que la extensión BOINC-MGE con sus dos estrategias de planificación disminuyen el consumo de batería promedio en los dispositivos y el tiempo total requerido para

finalizar un conjunto de trabajos bajo diferentes condiciones de batería limitada y escenarios de desconexión. Finalmente, la diferencia en el consumo de recursos de BOINC vs BOINC-MGE es despreciable, aunque el uso de recursos de BOINC-MGE es mayor.

Se ha logrado implementar en BOINC estrategias de planificación que tomen en cuenta características específicas de los dispositivos móviles. Estos resultados abren infinitas posibilidades para el uso de los teléfonos celulares inteligentes como dispositivos de cómputo para la solución de diferentes tipos de problemas, particularmente el procesamiento de imágenes.

10.1 Trabajo futuro

Teniendo en cuenta los prometedores resultados obtenidos en el presente trabajo de investigación, surgen varias posibilidades de mejora que pueden ser usadas o implementadas en futuros trabajos. Se describen a continuación algunas de ellas.

- Evaluar la extensión BOINC-MGE en una grid móvil real con un mayor número de dispositivos móviles donde se generen patrones más complejos de desconexiones y de consumo de batería en los dispositivos.
- Realizar la evaluación de las estrategias de planificación de BOINC-MGE con aplicaciones de mayor complejidad, es decir que presenten un mayor uso de la CPU y operaciones de entrada y salida, así como aplicaciones paralelas con dependencias entre trabajos.
- Implementar un mecanismo de puntos de control centralizado en BOINC que permita migrar las tareas de un dispositivo a otro, cuando el primero tenga problemas inminentes de batería. Este mecanismo sería la base para implementar en BOINC estrategias de tipo reactivas.
- Ampliar la API de desarrollo a un Kit de Desarrollo (SDK) que se encuentre disponible en lenguajes de programación diferentes a C++, lo cual facilitará aún más la implementación de nuevas estrategias de planificación en BOINC-MGE en forma de *plugins* que evite tener que compilar todos los componentes de BOINC por cada estrategia que se desee incorporar.
- Evaluar el rendimiento y comportamiento de BOINC-MGE en grids hibridas que incorporen además de dispositivos móviles, nodos de computo fijos como PC. A pesar de que en el diseño y desarrollo de BOINC-MGE se tuvo en cuenta este tipo de escenarios no se realizaron pruebas por estar fuera del alcance del proyecto.
- Dado que se ha probado la factibilidad de BOINC y de los dispositivos móviles para realizar cierto tipo de tareas de cómputo en paralelo, es tiempo de evaluar su uso en el campo de la salud, especialmente en el procesamiento de imágenes médicas. Algunas de las interrogantes a responder serían: ¿cuáles serían los casos de uso? ¿En qué tipo de instituciones de la salud sería conveniente esta infraestructura? ¿Para qué problemas específicos de imágenes médicas? ¿Es conveniente seguir con el modelo voluntario de BOINC para los problemas de salud que se seleccionen? ¿Cómo se integran las librerías de procesamiento de imágenes a BOINC?

Referencias

1. Mobile subscriptions outlook – Ericsson Mobility Report, https://www.ericsson.com/en/mobility-report/mobile-subscriptions-worldwide-outlook.
2. Anderson, D.P.: BOINC: A system for public-resource computing and storage. Proc. - IEEE/ACM Int. Work. Grid Comput. 4–10 (2004).
3. Litzkow, M.J., Livny, M., Mutka, M.W.: Condor - A hunter of idle workstations. In: Proceedings. The 8th International Conference on Distributed Distributed Computing Systems, 1988., 8th International Conference on. :104-111 1988. IEEE Conference (1988).
4. Datta, P., Dey, S., Paul, H.S., Mukherjee, A.: ANGELS: A framework for mobile grids. Proc. - Int. Conf. 2014 Appl. Innov. Mob. Comput. AIMoC 2014. 15–20 (2014).
5. Masinde, M., Bagula, A., Ndegwa, V.: MobiGrid: A middleware for integrating mobile phone and grid computing. Proc. 2010 Int. Conf. Netw. Serv. Manag. CNSM 2010. 523–526 (2010).
6. Palmer, N., Kemp, R., Kielmann, T., Bal, H.: Ibis for Mobility: Solving Challenges of Mobile Computing Using Grid Techniques. Proc. 10th Work. Mob. Comput. Syst. Appl. 17:1--17:6 (2009).
7. Hirsch, M., Mateos, C., Zunino, A.: Augmenting computing capabilities at the edge by jointly exploiting mobile devices: A survey. Futur. Gener. Comput. Syst. 88, 644–662 (2018).
8. Viswanathan, H., Lee, E.K., Pompili, D.: Mobile grid computing for data- and patient-centric ubiquitous healthcare. 2012 1st IEEE Work. Enabling Technol. Smartphone Internet Things, ETSIoT 2012. 36–41 (2012).
9. Rodríguez, J.M., Mateos, C., Zunino, A.: Are smartphones really useful for scientific computing? In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 38–47 (2012).
10. Duan, L., Kubo, T., Sugiyama, K., Huang, J., Hasegawa, T., Walrand, J.: Motivating smartphone collaboration in data acquisition and distributed computing. IEEE Trans. Mob. Comput. 13, 2320–2333 (2014).
11. Arslan, M.Y., Singh, I., Singh, S., Madhyastha, H. V., Sundaresan, K., Krishnamurthy, S. V.: Computing while charging: Building a Distributed Computing Infrastructure Using Smartphones. 8th Int. Conf. Emerg. Netw. Exp. Technol. 193–204 (2012).
12. Rafael, R., Escallón, G.: Grids Accesibles. Trab. grado. (2015).
13. Calle, D., Santamaría, A., Suárez, D.: Grid móvil para procesar imágenes médicas, (2016).
14. Curiel, M., Calle, D.F., Santamaría, A.S., Suarez, D.F., Flórez, L.: Parallel Processing of Images in Mobile Devices using BOINC.
15. Furthmüller, J., Waldhorst, O.P.: Survey on Grid Computing on Mobile Consumer Devices. In: Handbook of Research on P2P and Grid Systems for Service-oriented Computing: Models, Methodologies, and Applications. p. 25 (2001).
16. Litke, A., Skoutas, D., Varvarigou, T.: Mobile Grid Computing: Changes and Challenges of Resource Management in a Mobile Grid Environment.
17. Schaffner, B., Sawin, J., Myre, J.M.: Smartphones as Alternative Cloud Computing Engines: Benefits and Trade-offs. 2018 IEEE 6th Int. Conf. Futur. Internet Things Cloud. 244–250 (2018).
18. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations. Int. J. High Perform. Comput. Appl. (2001).
19. BOINC Wiki: VolunteerComputing – BOINC,

https://boinc.berkeley.edu/trac/wiki/VolunteerComputing.
20. Rodriguez, J.M., Zunino, A., Campo, M.: Introducing mobile devices into Grid systems: a survey. Int. J. Web Grid Serv. 7, (2011).
21. Sangolli, D.R., Ravindrarao, N.M., Patil, P.C., Palissery, T., Liu, K.: Enabling high availability edge computing platform. Proc. - 2019 7th IEEE Int. Conf. Mob. Cloud Comput. Serv. Eng. MobileCloud 2019. 85–92 (2019).
22. Curiel, M.: Wireless Grids: Recent Advances in Resource and Job Management. (2018).
23. Li, Z., Sun, L., Ifeachor, E.C.: Challenges of Mobile ad-hoc Grids and their Applications in e-Healthcare. (2015).
24. Ltd., A.P.C.P.: Decision Analysis and Resolution Guidelines.
25. Highsmith, J.: Adaptive software development: a collaborative approach to managing complex systems. Addison-Wesley (2013).
26. Kesselman, C., Tuecke, S.: the Anatomy of the Grid : Enabling Scalable Virtual Organizations. 6–7 (2001).
27. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The physiology of the grid: An open grid services architecture for distributed systems integration. Open Grid Serv. Infrastruct. WG, Glob. Grid Forum. 22, 2002 (2002).
28. Foster, I., Kesselman, C.: Globus: A metacomputing infrastructure toolkit. Int. J. High Perform. Comput. Appl. (1997).
29. Anderson, B.D.P., Cobb, J., Korpela, E., Lebofsky, M.: SETI@home An experiment in public-resource computing. 45, (2002).
30. Kurdi, H., Li, M., Al-Raweshidy, H.: A Classification of Emerging and Traditional Grid Systems. 9, (2008).
31. Lee, J., Song, S.J., Gil, J., Chung, K., Suh, T., Yu, H.: Balanced scheduling algorithm considering availability in mobile grid. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 211–222 (2009).
32. Vaithiya, S.S., Bhanu, S.M.S.: Scheduling tasks in mobile grid environment using mobility based resource prediction. 2010 1st Int. Conf. Parallel, Distrib. Grid Comput. PDGC - 2010. 89–94 (2010).
33. Rodriguez, J.M., Zunino, A., Campo, M.: Mobile Grid SEAS : Simple Energy-Aware Scheduler. 54, 3341–3354 (2010).
34. Hirsch, M., Rodriguez, J.M., Zunino, A., Mateos, C.: Battery-aware centralized schedulers for CPU-bound jobs in mobile Grids. Pervasive Mob. Comput. 29, 73–94 (2016).
35. Lee, D.W., Chin, S.H., Gil, J.M.: Efficient resource management and task migration in mobile grid environments. Commun. Comput. Inf. Sci. 78 CCIS, 384–393 (2010).
36. Adeyelu, A., Olajubu, E., Aderounmu, A., Ge, T.: A Model for Coordinating Jobs on Mobile Wireless Computational Grids. Int. J. Comput. Appl. 84, 17–24 (2013).
37. BOINC Wiki: Android FAQ - BOINC, https://boinc.berkeley.edu/wiki/Android_FAQ.
38. Basic Concepts – BOINC, https://boinc.berkeley.edu/trac/wiki/BasicConcepts.
39. Anderson, D.P., Reed, K.: Celebrating diversity in volunteer computing. Proc. 42nd Annu. Hawaii Int. Conf. Syst. Sci. HICSS. 1–8 (2009).
40. Taufer, M., Anderson, D., Cicotti, P., Brooks, C.L.: Homogeneous redundancy: A technique to ensure integrity of molecular simulation results using public computing. Proc. - 19th IEEE Int. Parallel Distrib. Process. Symp. IPDPS 2005. 2005, (2005).
41. BOINC Wiki - The BOINC application programming interface (API),

https://boinc.berkeley.edu/trac/wiki/BasicApi.
42. Du, L.J., Yu, Z.W.: Scheduling algorithm with respect to resource intermittence in mobile grid. 2010 6th Int. Conf. Wirel. Commun. Netw. Mob. Comput. WiCOM 2010. 2–6 (2010).
43. Huang, Y., Mohapatra, S., Venkatasubramanian, N.: An energy-efficient middleware for supporting multimedia services in mobile grid environments. ITCC 2005 Int. Conf. Inf. Technol. Coding Comput. Vol 2. 220–225 (2005).
44. AdaptiveReplication – BOINC, https://boinc.berkeley.edu/trac/wiki/AdaptiveReplication.
45. Huang, C.Q., Zhu, Z.T., Wu, Y.H., Xiao, Z.H.: Power-aware hierarchical scheduling with respect to resource intermittence in wireless grids. Proc. 2006 Int. Conf. Mach. Learn. Cybern. 2006, 693–698 (2006).
46. Chin, S.H., Suh, T., Yu, H.C.: Genetic algorithm based scheduling method for efficiency and reliability in mobile grid. Proc. 4th Int. Conf. Ubiquitous Inf. Technol. Appl. ICUT 2009. 1–6 (2009).
47. Li, C., Li, L.: Energy constrained resource allocation optimization for mobile grids. J. Parallel Distrib. Comput. 70, 245–258 (2010).
48. Mcgough, A.S., Forshaw, M.: Evaluation of Energy Consumption of Replicated Tasks in a Volunteer Computing Environment. 85–90 (2018).
49. Alenawy, T.A., Aydin, H.: Energy-constrained scheduling for weakly-hard real-time systems. Proc. - Real-Time Syst. Symp. (2005).
50. Xie, T., Qin, X., Nijim, M.: Solving energy-latency dilemma: Task allocation for parallel applications in heterogeneous embedded systems. Proc. Int. Conf. Parallel Process. 12–19 (2006).
51. Kim, K.H., Buyya, R., Kim, J.: Power aware scheduling of bag-of-tasks applications with deadline constraints on DVS-enabled clusters. Proc. - Seventh IEEE Int. Symp. Clust. Comput. Grid, CCGrid 2007. 541–548 (2007).
52. Liang, W.-Y., Lai, P.-T., Chiou, C.W.: An Energy Conservation DVFS Algorithm for the Android Operating System. (2010).
53. Canny, J.: A Computational Approach to Edge Detection. IEEE Trans. Pattern Anal. Mach. Intell. (1986).
54. Online: CannyEdgeDetector C++ implementation, https://github.com/resset/CannyEdgeDetector.
55. Stoop, W.: Bitmap API - C implementation, https://github.com/wernsey/bitmap.
56. Jain, R.: The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling. John Wiley & Sons (1990).