Tesis defendida por

Aritz Barrondo Corral

y aprobada por el siguiente Comité

Dr. Andrey Chernykh

Director del Comité

Dr. Carlos Alberto Brizuela Rodriguez

Miembro del Comité

Dr. Jesús Favela Vara

Miembro del Comité

Dr. Rodger Terence Evans

Miembro del Comité

Dra. Elisa Schaeffer

Miembro del Comité

Dr. José Antonio García Macías

Coordinador del programa de posgrado en Ciencias de la Computación

---

Dr. David Hilario Covarrubias Rosales

Director de Dirección de Estudios de Posgrado

14 de Septiembre de 2012

CENTRO DE INVESTIGACIÓN CIENTÍFICA Y DE EDUCACIÓN SUPERIOR DE ENSENADA

Programa de posgrado en ciencias

en Ciencias de la Computación

Eficiencia Energética de Algoritmos de Calendarización para Grids de Computadoras Personales Punto-a-Punto

Tesis

que para cubrir parcialmente los requisitos necesarios para obtener el grado de

Maestro en Ciencias

Presenta:

Aritz Barrondo Corral

Ensenada, Baja California, México.

2012

Resumen de la tesis de **Aritz Barrondo Corral**, presentada como requisito parcial para la obtención del grado de Maestro en Ciencias en Ciencias de la Computación. Ensenada, Baja California, Septiembre de 2012.

Eficiencia Energética de Algoritmos de Calendarización para Grids de Computadoras Personales Punto-a-Punto

Resumen aprobado por:

Dr. Andrey Chernykh

Director de Tesis

Los Grids de Computadoras Personales Punto-a-Punto (P2P-DG, por sus siglas en inglés) son infraestructuras de cómputo que aglomeran grandes cantidades de computadoras personales. En este trabajo ataca un problema de optimización de estos recursos a través de calendarización de Bolsas de Trabajos (BoT, por sus siglas en inglés) no-preentiva en P2P-DG. Se consideran dos escenarios: en el primero, las decisiones de calendarización son hechas libres de información, y en el segundo sólo se basan en información de las características de las máquinas. Se considera un modelo de calendarización con replicación de tareas para sobrellevar una posible mala colocación y asegurar buen desempeño. Se incorpora un modelo de energía para evaluar el consumo de energía de los diferentes escenarios. Se proponen algoritmos conscientes del uso de energía y se hace una evaluación de desempeño usando un modelo simple que se enfoca en algunos de los aspectos más importantes de los P2P-DG, donde los servidores tienden a tener diferentes velocidades de procesamiento y eficiencia energética.

Palabras Clave: Punto-A-Punto, Grids de Computadoras Personales, Calendarización Libre de Información, Calendarización Consciente de la Energía, Cómputo Verde.

Abstract of the thesis presented by **Aritz Barrondo Corral**, in partial fulfillment of the requirements of the degree of Masters in Sciences in Computer Science. Ensenada, Baja California, September 2012.

Energy Efficiency of Scheduling Algorithms for Peer-to-Peer Desktop Grids

The Peer-to-Peer Desktop Grids (P2P-DG) are computer infrastructures that agglomerate large quantities of desktop computers. We address a resource optimization problem through non-preemptive Bag-of-Tasks (BoT) scheduling on P2P-DG. Two scenarios are considered: in the first one, scheduling decisions are taken knowledge free and, in the second one, they are based only on the knowledge of resource characteristics. We consider a scheduling model with task replication to overcome a possible bad allocation and ensure good performance. We incorporate an energy model in order to evaluate grid energy consumption in different scenarios. We propose power-aware scheduling algorithms and conduct a performance evaluation using a simple model that focuses on some key aspects of P2P-DG, where servers tend to have different processing speed and energy efficiency.

Keywords: Peer-to-Peer, Desktop Grid, Knowledge-Free Scheduling, Energy-Aware Scheduling, Green Computing.

Que este trabajo sea de beneficio para todos los seres.

Agradecimientos

A mi mamá, Psic. Clara Patricia Corral, a mi pareja Padma K. Beamonte y al resto de mi familia por su apoyo. Al Dr. Andrey Chernykh por su tiempo, ayuda y consejo, así como a todos los investigadores, estudiantes y personal del departamento de ciencias computacionales por su enseñanza académica.

Finalmente al CONACyT y proyecto de investigación por su apoyo económico.

Contenido

Página

Contenido (continuación)

Página

Lista de Figuras

Lista de Figuras (continuación)

Lista de Tablas

Lista de Tablas (continuación)

Capítulo 1

Introducción

"En el principio sólo estaba el mainframe, una computadora central que se encargaba de las tareas de proceso de los usuarios, que si no era buena, por lo menos era sencilla. Los usuarios y el personal por igual sabían dónde ocurrían las entradas, salidas y el procesamiento de datos. Este mainframe era el núcleo denso y caliente que se convertiría en el universo de cómputo que conocemos ahora. En los momentos justo después del "Big Bang" de la computación, el procesamiento en tiempo compartido y las terminales remotas separaron la entrada y salida de datos de su procesamiento, pero aún era posible ir a un cuarto y señalar a la caja donde se estaba haciendo el trabajo. Desde la década de 1960, científicos de la computación como John McCarthy previeron que los servicios de cómputo serían tratados como utilidades públicas. Una década más tarde, las redes y el Internet continuaron haciendo del procesamiento de datos una pieza cada vez más abstracta dentro del mundo de la computación" - Chee y Franklin (2010), p. 2.

Como parte del proceso de separación de la entrada y salida (E/S) de información y el procesamiento de datos basado en abstracción de recursos físicos, nacieron los grids computacionales, un conjunto de computadoras conectadas a través de distintas redes, que brindan la capacidad de procesamiento y almacenaje de este conjunto como si fueran una sola computadora. *"La computación en grid describe el uso de recursos geográficamente distribuidos para la solución de problemas de forma coordinada en organizaciones virtuales. Aunque los grids no están limitados a recursos computacionales y pueden incorporar diversos servicios y aparatos, un gran número de las instalaciones de grid son de Cómputo de Alto Desempeño (HPC, por sus siglas en inglés) que in-*

cluyen computadoras paralelas de distintos tamaños" Schwiegelshohn *et al.* (2008), p. 1061. Es por esto, que creciente número de disciplinas científicas colaboran en grids a través de organizaciones virtuales para atacar problemas de investigación complejos de forma conjunta y compartir recursos computacionales (Tchernykh *et al.*, 2010).

El término grid hace analogía a los grids de energía eléctrica, que dan de forma transparente al usuario, acceso a recursos originados en distintas plantas de energía (Foster y Kesselman, 1999). Además, los grids computacionales y eléctricos han seguido una evolución similar. Hoy en día, los grids eléctricos, en lugar de estar compuestos sólo por plantas de energía relativamente grandes, permiten a los individuos producir su propia energía e inyectar a la red pública su excedente, contribuyendo de esta forma con los recursos de la comunidad. Los grids computacionales llevan el mismo tipo de evolución, donde computadoras individuales pueden contribuir con un grid, ya sea total o parcialmente, y posiblemente por espacios de tiempo limitado. Esto significa, que los grids computacionales ya no pueden ser considerados sólo como una federación modesta de recursos computacionales poderosos. Al contrario, las tendencias muestran que los grids del futuro estarán compuestos de grandes cantidades de computadoras individuales, las cuales, pueden o no, ser parte de recursos computacionales poderosos (Pierre, 2007).

1.1 Grids de Computadoras Personales

Las plataformas típicas de grids computacionales por lo general están compuestas de relativamente pocos, pero muy poderosos y costosos recursos; pequeños laboratorios que no pueden costear dichos recursos ni el personal para mantenerlos, no pueden utilizar la computación en grid para satisfacer sus necesidades de cómputo. Una alternativa

de bajo costo, tanto en la academia como en la industria, ha sido el usar grids de computadoras personales. Este tipo de grids une un gran número de computadoras de escritorio que pueden pertenecer a individuos independientes (Cirne *et al.*, 2006; Anglano *et al.*, 2010). De igual manera, estos sistemas tienen distintas variantes, las cuales se clasifican según cuatro características básicas (Choi *et al.*, 2007):

1) Organización, que puede ser centralizada o distribuida. Ejemplos de grids de computadoras personales centralizados, está SETI@Home (Anderson *et al.*, 2002) y otros derivados de BOINC (Anderson, 2004); descentralizados, está Organic Grid (Chakravarti *et al.*, 2004), OurGrid (Cirne *et al.*, 2006) y ShareGrid (Anglano *et al.*, 2010) que son sistemas Punto-a-Punto (P2P, por sus siglas en inglés).

2) Plataforma, que puede ser basada en Web o en algún middleware. Entre los basados en Web existen grids que distribuyen trabajos a través de applets de Java que corren en navegadores web, como lo es Javelin (Neary *et al.*, 2000) y WebCom-G (Morrison *et al.*, 2005); mientras entre los basados en middleware están OurGrid, ShareGrid , BOINC y Organic Grid .

3) Escala, que distingue entre los de escala global a través de Internet o local por medio de una red de área local (LAN, por sus siglas en inglés). Algunos ejemplos de grids a escala de Internet son ShareGrid, BOINC, Organic Grid; a escala de una LAN está Condor (Pierre, 2007) que forma la base para grids empresariales.

4) Proveedor de recursos, que pueden ser voluntarios, compartidos o comerciales. Algunos ejemplos de proveedores de recursos computacionales voluntarios son ShareGrid, BOINC, Organic Grid, mientras que entre los comerciales está Condor y algunas versiones de Entropía (Chien *et al.*, 2003).

La diferencia principal entre los recursos compartidos y voluntarios es que en un grid compartido, cualquiera puede enviar un trabajo y en los grids voluntarios, se ejecuta un proyecto centralizado. Además, los grids voluntarios tienden a estar compuestos por máquinas particulares, a diferencia de los grids compartidos que suelen contener máquinas correspondientes a instituciones de investigación(Ponciano y Brasileiro, 2010). Por lo que en el primero se deben gastar recursos, que muchos laboratorios no tienen, en difusión del grid para obtener la visibilidad requerida para formar el grid de voluntarios. De igual manera, estos grids, para poder proveer un beneficio en cómputo, tolerancia a fallas y heterogeneidad, deben crecer bastante. Más aparte, el mantener un servidor central implica otros costos. Por lo que este tipo de grid sólo funciona en instituciones reconocidas, como lo es la Universidad de Berkeley, la cual maneja y promueve BOINC (Anglano *et al.*, 2010).

1.2 Calendarización en Grids de Computadoras Personales Punto-a-Punto

Los P2P-DG son los que cuentan con una organización distribuida, están basados en middleware y los recursos los proveen voluntarios a través de todo Internet. Este tipo de grid necesita ser rápido, simple, escalable y seguro. Pero debido a que la conexión a Internet no es universal (debido a direccionamiento privado y corta fuegos) y que nuevas vulnerabilidades aparecen diario, lograr estos objetivos es muy difícil. Para simplificar el problema, algunos grids han reducido su alcance al soportar únicamente aplicaciones de Bolsa-de-Tareas (BoT, por sus siglas en inglés) (Cirne *et al.*, 2006).

Las aplicaciones BoT son aplicaciones paralelas cuyas tareas son independientes. De igual forma, las BoT simplifican los requerimientos ya que tareas fallidas no afectan otras tareas, por lo que el grid puede ofrecer una ejecución rápida sin pedir garantías

de calidad de servicio a los recursos. También facilitan el proveer un ambiente seguro, debido a que no se necesita acceso a la red durante la ejecución de una tarea foránea. A pesar de su simplicidad, aplicaciones BoT son usadas en gran variedad de escenarios, que incluyen minería de datos, búsquedas masivas, simulaciones, ecuaciones fractales, biología computacional y procesado de imágenes (Silva *et al.*, 2003; Cirne *et al.*, 2006; Anglano *et al.*, 2010).

La calendarización es otro aspecto que simplifican las aplicaciones BoT. Las aplicaciones BoT pueden ser calendarizadas con algoritmos libres de información, los cuales no requieren información de las tareas ni de las máquinas que las procesan. Un ejemplo de este tipo de algoritmo es *Work Queue with Replication* (WQR), el cual manda las tareas a las máquinas de forma aleatoria; cuando ya no quedan tareas por mandar, aleatoriamente se replica alguna de la tareas en ejecución; y cuando una máquina termina una tarea, sus réplicas se cancelan y se le manda otra tarea, como se ilustra en la Figura 1. Las réplicas son la clave para recuperarse de malas asignaciones de tareas en máquinas (las cuales son inevitables dada la aleatoriedad). El desempeño de WQR es tan bueno como el de los algoritmos de calendarización tradicionales basados en el uso de información, y a los cuales se les alimenta con información perfecta (Silva *et al.*, 2003). Sin embargo, consume más ciclos de proceso, los cuales se traducen directamente a consumo de energía (Barrondo *et al.*, 2012).

1.3 Calendarización Consciente de Energía

Reducir el consumo de energía se ha convertido en uno de los mayores problemas a los que se ha enfrentado la industria de la computación, tanto por la cantidad de dinero que se gasta para mantener funcionando las computadoras (Kurp, 2008; Valentini *et al.*,

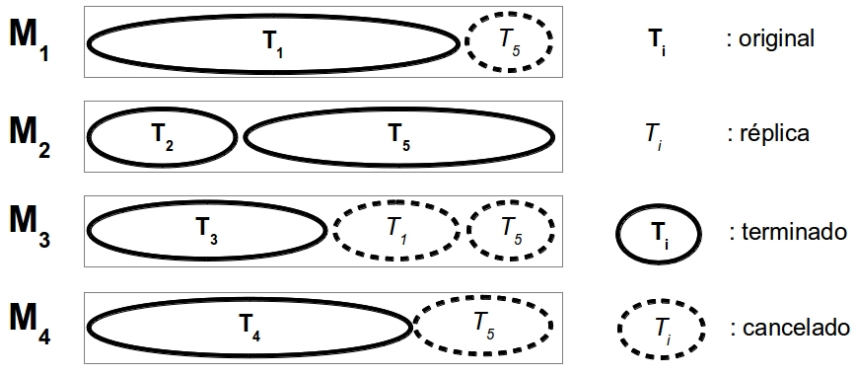

Figura 1. Cola de Trabajos con Replicación.

2011) y la reducción en la confiabilidad de los sistemas al aumentar la temperatura (Ge *et al.*, 2005b; Kim *et al.*, 2007; Vasić *et al.*, 2009), como por el impacto que los generadores de energía tienen sobre el medio ambiente.

Existen dos estrategias para hacer un calendario consciente de energía. La primera, el Escalamiento Dinámico de Voltaje (DVS, por sus siglas en inglés), ha sido ampliamente estudiado, y ha dado buenos resultados en distintos escenarios (Kim *et al.*, 2007; Sharma y Aggarwal, 2009). La segunda, la desactivación dinámica de componentes (DCD, por sus siglas en inglés) según las cargas de trabajo (Develder *et al.*, 2008; Barrondo *et al.*, 2012).

El problema de las estrategias DVS es que se requiere información confiable (Valentini *et al.*, 2011), tanto del grid como de las tareas. Esta información es difícil de obtener en ambientes heterogéneos y dinámicos (Silva *et al.*, 2003; Orgerie *et al.*, 2010) como los son los P2P-DGs (Cirne *et al.*, 2006; Anglano *et al.*, 2010).

1.4 Definición del Problema

En esta tesis se resuelve un problema de optimización de recursos a través de dos objetivos. El primero, es minimizar el tiempo de terminación de la carga de trabajo

o conjunto de tareas. Este objetivo se puede considerar como tiempo de respuesta o calidad de servicio (QoS, por sus siglas en ingles). El segundo objetivo, es minimizar el consumo de energía. Este objetivo representa el gasto de operación de las máquinas al ejecutar la carga de trabajo.

1.5 Objetivo

El objetivo general de esta tesis es analizar los diferentes algoritmos de calendarización para P2P-DGs, evaluar su eficiencia energética y recomendar un algoritmo que es energéticamente eficiente y mantiene la calidad de servicio. Para ésto se definieron una serie de objetivos específicos y preguntas de investigación.

1.5.1 Objetivos específicos

- Estudiar modelos de energía para PCs.
- Implementar un modelo de energía para P2P-DGs.
- Diseñar un ambiente de simulación.
- Evaluar la eficiencia energética de los algoritmos de calendarización para Grids de P2P-DGs.
- Diseñar algoritmos de calendarización energéticamente eficientes para Grids de P2P-DGs.

1.5.2 Preguntas de investigación

- ¿Cómo evaluar el consumo de energía de un grid de computadoras personales utilizando distintos algoritmos de calendarización?

- ¿Cómo benefician o afectan al desempeño los distintos algoritmos?
- ¿Cómo benefician o afectan al consumo de energía los distintos algoritmos?
- ¿Cuál es la relación entre consumo de energía y desempeño?
- ¿Qué algoritmo usar para minimizar el consumo de energía manteniendo límites de desempeño?

1.6 Justificación

Como muestran las tendencias (Anderson *et al.*, 2002; Silva *et al.*, 2003; Anderson, 2004; Anglano y Canonico, 2005; Cirne *et al.*, 2006; Choi *et al.*, 2007; Anglano y Canonico, 2008; Cunsolo *et al.*, 2009; Anglano *et al.*, 2010; Ponciano y Brasileiro, 2010; Barrondo *et al.*, 2012; Awan y Jarvis, 2012), se están desarrollando P2P-DGs de bajo costo para ayudar con el procesamiento de datos de proyectos científicos. Pero debido a la dificultad de obtener buena información en grids tan dinámicos y heterogéneos (Silva *et al.*, 2003), como lo son los P2P-DG, no se han implementado estrategias de manejo de energía que aseguren un buen desempeño en el calendario. Como resultante, los donadores de recursos computacionales están desperdiciando más energía de la que deberían, aún cuando no apaguen sus computadoras para poder donar recursos (Barrondo *et al.*, 2012). Este desperdicio tiene un efecto negativo tanto en la economía del proveedor de recursos, así como en el medio ambiente (Kurp, 2008).

Además, es factible ofrecer servicios de cómputo en nube usando los recursos de un P2P-DG (Cunsolo *et al.*, 2009). Esto transforma a los donadores de recursos en proveedores de recursos comerciales. En este escenario, el implementar estrategias de manejo de energía reduciría los costos de proveer recursos computacionales e incrementaría las ganancias de los proveedores. En este trabajo se incorpora el consumo de

energía al esquema de WQR. Se evalúan los algoritmos de calendarización libres de información actuales y se determina cómo el nivel de información usado para seleccionar las máquinas para la asignación de tareas afecta el desempeño y el consumo de energía.

1.7 Organización de la Tesis

El resto de la tesis está organizada de la siguiente manera. En el siguiente capítulo (Capítulo 2), se presenta el estado del estado del arte y distintos problemas abiertos en P2P-DGs, manejo de energía y calendarización. En el Capítulo 3, la metodología, se describen los modelos de Grid, de Energía, Carga de Trabajo, Estrategias de Calendarización y método de evaluación usados en los Experimentos. En el Capítulo 4 se presenta el diseño experimental y resultados del primer conjunto de experimentos (WQR), donde se replica el trabajo de la bibliografía (Silva *et al.*, 2003) añadiendo el modelo de energía. En el Capítulo 5 se presenta el diseño experimental y resultados del segundo conjunto de experimentos (WQR-x), el cual extiende el diseño experimental anterior añadiendo estrategias de calendarización con distintos niveles de información. En el Capítulo 6 se concluye, seguido de las referencias y los apéndices.

Capítulo 2

Estado del Arte

En este capítulo se presenta una breve revisión del estado del arte de los P2P-DG así como de las distintas estrategias de calendarización y de manejo de energía, resaltando los aspectos relevantes para los problemas a tratar en el resto de la tesis.

Aunque los P2P-DG ya son una realidad, existen una variedad de problemas a resolver, como lo son qué topología usar y por qué motivos, la transferencia eficiente de información y la calendarización de tareas. También existen propuestas para ofrecer servicios de más alto nivel como redes sociales (Marcon *et al.*, 2011), o un mercado de compra/venta máquinas virtuales abierto, pero en vez de usar centros de datos, usa grids de computadoras personales (Cunsolo *et al.*, 2009).

La calendarización ha sido una de las áreas de investigación con mayor relevancia en el desarrollo del cómputo distribuido. Ahora el ahorro de energía en las computadoras que proveen los recursos de cómputo físicos también se ha convertido en uno de los problemas principales de la computación actual (Kurp, 2008). Por lo que se resumen las estrategias usadas en los distintos tipos de sistemas de cómputo distribuido incluyendo la calendarización.

2.1 Grids de Computadoras Personales

En esta sección se presenta el estado del arte así como distintos problemas abiertos para este tipo de sistemas distribuido.

2.1.1 Topologías

La forma en que se organiza un grid tiene repercusiones directas en el descubrimiento y manejo de los recursos que lo integran, así como la escalabilidad que tolere el sistema (Miriam y Easwarakumar, 2011). Existen varias propuestas de *Topologías*, como el Hiper-árbol, que por la naturaleza de árbol, es centralizado (Goodman y Sequin, 1981). También se ha propuesto una topología de Hiper-Cubo para grids P2P debido a su descentralización (Miriam y Easwarakumar, 2011). El manejo de recursos es hecho por todas las computadoras, pero cada una sólo maneja la información de un pequeño número de vecinos, representa a esos vecinos ante otras computadoras y actúa como enrutador de las peticiones de recursos de las computadoras del grid, al igual que como proveedor de servicios y consumidor de los mismos. Esta propuesta es considerablemente escalable en términos de tiempo, ya que mantiene un número máximo de saltos para la resolución de una petición. Por otra parte, brinda tolerancia a fallos gracias al gran número de conexiones que tiene cada computadora.

Otros sistemas han adoptado un manejo libre de información, como es el caso de ShareGrid. Que no presentan una topología determinada, en cambio se lleva un registro de las máquinas conectadas, pero todas éstas se comunican de forma P2P, como se muestra en la Figura 2, donde cada recurso físico y usuario tienen un representante virtual que se inscribe en un registro central, pero puede tener comunicación punto a punto con los demás representantes virtuales (Anglano *et al.*, 2010).

2.1.2 Transferencia de datos

Programas que corren en este tipo de grid, requieren grandes cantidades de información. En las arquitecturas centralizadas, como BOINC (Anderson, 2004), este requerimiento

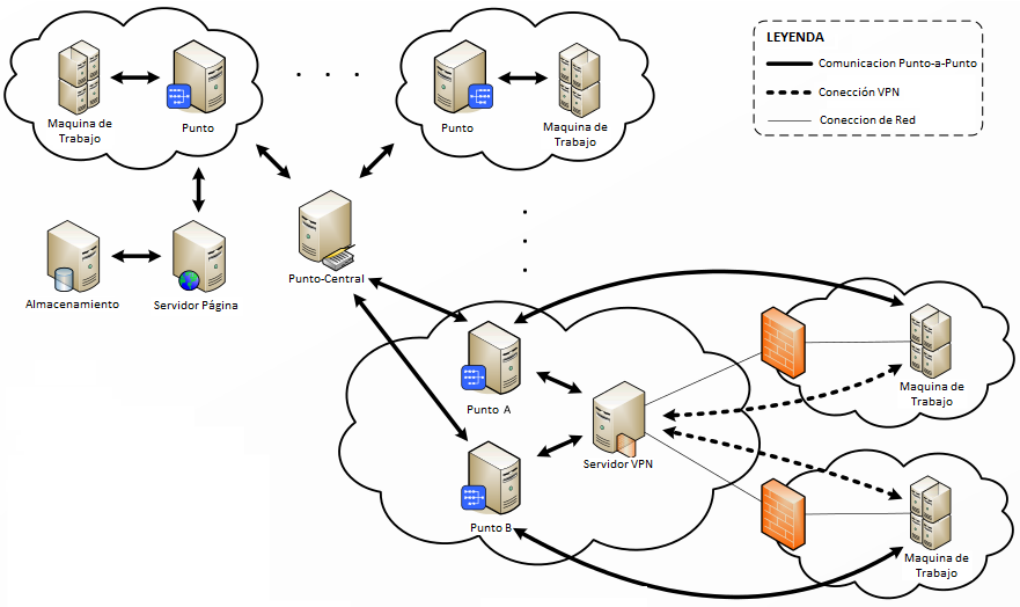

Figura 2. Arquitectura de ShareGrid, Anglano et al., 2010, p. 549.

puede causar cuellos de botella cuando los archivos con esta información son muy grandes, o el servidor que distribuye la información tiene ancho de banda limitado. Se modificó BOINC al sustituir los servidores HTTP, que distribuyen información por todo el grid, por redes tipo Bit Torrent, redes de distribución de contenido (CDN, por sus siglas en inglés) o algún otro tipo de red P2P para la distribución de información. La idea principal de este cambio de protocolo es buscar la colaboración en la distribución de información de todas las máquinas conectadas. El estudio demostró que Bit Torrent es una buena alternativa para la distribución de archivos grandes, pero no es eficiente distribuyendo archivos de menor tamaño, por lo que se propone el uso de FTP para estos últimos (Costa *et al.*, 2008).

Existen otras estrategias para el manejo de la información dentro de un grid de computadoras personales. En OurGrid, se usa un parámetro de afinidad de almacenaje. Éste trata de explotar la reutilización de información y evitar transferencias innecesarias (Cirne *et al.*, 2006).

2.1.3 Nube en casa

Una aplicación poco estudiada pero con gran potencial en el uso de los grids de computadoras personales para brindar los recursos de cómputo físicos para una infraestructura de nube (Cunsolo *et al.*, 2009; Chee y Franklin, 2010). Como se menciona en Weiss (2007), las nubes (haciendo referencia a Cloud Computing) tienen distintas formas según la perspectiva con que se vean. El cómputo en nube es un modelo de procesamiento de información en el cual capacidades de cómputo son entregadas en forma de servicios, conforme se van necesitando, a través de Internet, a una variedad de dispositivos de usuarios (Chee y Franklin, 2010). Estas capacidades pueden ser provistas por un grid (Murphy *et al.*, 2010).

Se propuso un sistema de grid de computadoras personales centralizado que brinde soporte a una infraestructura de nube (Cunsolo *et al.*, 2009). Dando como resultado una nube más democrática donde los usuarios tienen más poder, y no sólo son consumidores, sino también proveedores de servicios. Se dan dos posibilidades de provisión de recursos, voluntarios que donen capacidades de cómputo a proyectos científicos, y un mercado donde los usuarios puedan vender las mismas (Cunsolo *et al.*, 2009; Chee y Franklin, 2010). Se identifican seis retos por superarse antes de que ésta visión se pueda hacer realidad:

1. Manejo de recursos y servicios: se sugiere la implementación de algún mecanismo que automatice el descubrimiento y la administración de los recursos que componen el grid y los servicios que ofrece la nube (Schwiegelshohn y Tchernykh, 2012).

2. Abstracciones para la interfaz de usuario: se debe proveer al usuario un punto de vista orientado a servicios así como un punto de acceso único y uniforme a la

nube.

3. Seguridad: los mecanismos de seguridad deben proveer autenticación de usuarios, protección de recursos e información y confiabilidad e integridad de la información.

4. Fiabilidad: se requieren mecanismos que le den tolerancia a fallas al sistema, como lo es redundancia de recursos y servicios y políticas de recuperación de servidores.

5. Inter-operabilidad: para poder comunicarse y operar junto con otros sistemas, y dada la naturaleza abierta de un grid de computadoras personales, las nubes en P2P-DG pueden ayudar al desarrollo de un estándar para el cómputo en nube que solucione el problema de bloqueo de información que advirtió Richard Stallman en una entrevista (Johnson, 2008) y es más ampliamente discutido en Armbrust *et al.* (2009).

6. Modelo de negocio: es necesario poder proveer algún tipo de calidad de servicio o acuerdo para poder diferenciar entre aplicaciones a ser ejecutadas.

Se puede discutir que el desarrollo de sistemas de cómputo en nube con proveedores de servicios descentralizados puede traer una nueva era de computación personal alejándose de las tendencias actuales pro-centralización que se tiene en la industria de la computación actual.

2.2 Manejo de Energía

Desde 1992, el desempeño de sistemas de alto desempeño (HPC, por sus siglas en inglés) como supercomputadoras, se ha duplicado 10000 veces a diferencia de la eficiencia energética de los mismos, la cual sólo se ha duplicado 300 veces (Feng y Cameron,

2007). Debido a esta falta de eficiencia energética, las tecnologías de información verdes (*Green IT*) han surgido como un nuevo dominio de investigación.

A continuación se describen las estrategias de manejo de energía usados en los tres paradigmas principales de sistemas de cómputo distribuido (Valentini *et al.*, 2011):

(a) Clusters Computacionales, que consta de la integración de computadoras comerciales que incorpora hardware, redes y software para crear una imagen única del sistema.

(b) Grids Computacionales, que consta de un sistema de recursos computacionales heterogéneos en distintos dominios administrativos, que trabajan juntos para alcanzar una meta común.

(c) Nubes Computacionales, que consta de un modelo basado en Internet de entrega y consumo de servicios de cómputo bajo demanda.

A continuación se describe el estado del arte en el manejo de energía en cada uno de ellos, haciendo énfasis en los grids de computadoras personales.

2.2.1 Manejo de energía en clusters

Los *Clusters Computacionales* consisten en un grupo de computadoras independientes conectadas por una red de alta velocidad dedicada. En estos sistemas, la complejidad interna se esconde del usuario a través del uso de *middleware* (Buyya *et al.*, 2001). Hoy en día, estos sistemas son una alternativa competitiva al cómputo de alto desempeño tradicional (Valentini *et al.*, 2011).

Como resultado se tiene un excesivo consumo de energía y densidad de componentes que producen grandes cantidades de calor, junto con una creciente demanda de electricidad. Por lo que las dos grandes áreas de preocupación dentro de los Clusters son : (a)

el incremento en los costos operacionales y (b) reducción de la confiabilidad del sistema (Ge *et al.*, 2005b; Kim *et al.*, 2007; Vasić *et al.*, 2009), debido a que el cómputo a altas temperaturas es más propenso a fallas que el cómputo a temperaturas moderadas (Ge *et al.*, 2005b; Kim *et al.*, 2007).

La eficiencia energética de un sistema Cluster puede ser mejorada en tres niveles (Beloglazov *et al.*, 2010): (a) aplicaciones energéticamente eficientes, (b) manejo de recursos consciente de energía y (c) hardware energéticamente eficiente.

El consumo total de energía (E) en un cluster en un intervalo de tiempo (T) se define como el producto del tiempo (T) y la potencia (P) requerida por el sistema, como se muestra en la ecuación 1,

$$E = T \cdot P. \tag{1}$$

Por lo que sólo se puede reducir el consumo reduciendo la energía requerida (P) o el intervalo de tiempo (T). Usualmente, las limitaciones en E son el resultado del mapeo de aplicaciones a clusters individuales, por lo que el manejo de la potencia requerida se hace incrementalmente importante (Valentini *et al.*, 2011).

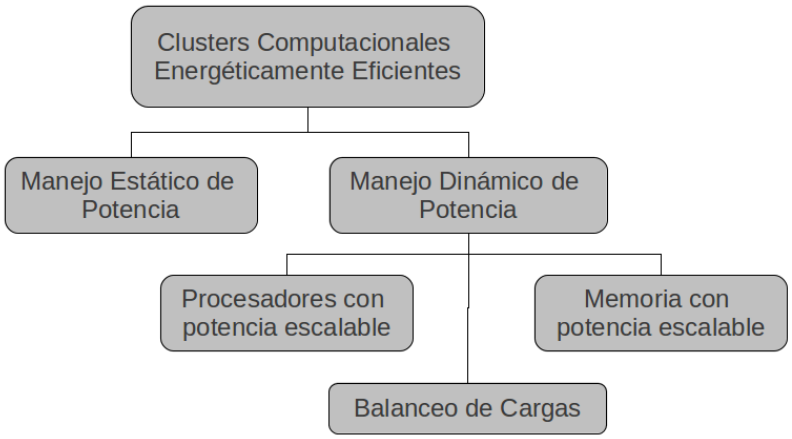

Figura 3. Manejo de potencia en Clusters Computacionales, adaptado de Valentini et al., 2011, p. 4.

Como se muestra en la Figura 3, existen dos categorías principales en el manejo de la potencia requerida en los Clusters Computacionales:

(a) Manejo Estático de Potencia (SPM, por sus siglas en inglés), el cual consiste en el uso de componentes energéticamente eficientes.

(b) Manejo Dinámico de Potencia (DPM, por sus siglas en inglés), el cual usa información del estado de los recursos para reducir el consumo de energía.

SPM ha sido usando efectivamente en la última década en sistemas de mano y móviles (Ge *et al.*, 2005a). Posteriormente fue implementado para el sistema *Fast Array of Wimpy Nodes* (FAWN) mostrando que los CPUs con frecuencia más baja son más eficientes que los CPUs de alto desempeño (Andersen *et al.*, 2009). Y para la arquitectura "Gordon", el cual es un ejemplo de un sistema centrado en datos y bajo requerimiento de potencia. Al utilizar procesadores de bajo consumo y memoria flash, supera de 1.5 a 2.5 veces en desempeño por Watt a clusters basados en discos (Caulfield *et al.*, 2009). La desventaja de esta técnica es que los componentes energéticamente eficientes tienden a tener mayor costo.

La técnica de DPM se ha mostrado prometedora para el manejo de energía, pero el diseño de algoritmos de calendarización conscientes de los requerimientos de potencia, no es trivial (Valentini *et al.*, 2011). Como se muestra en la Figura 3, DPM usa por una parte software y componentes con requerimientos de potencia escalables para hacer ajustes dinámicos al consumo de energía (Beloglazov *et al.*, 2010; Ge *et al.*, 2005b,a; Tolentino *et al.*, 2007), así como técnicas de balanceo de cargas (Pinheiro *et al.*, 2001).

Los clusters computacionales que utilizan software y componentes con potencia escalable son llamados *Clusters Conscientes de la Potencia* (Ge *et al.*, 2005a). Éstos sistemas se subdividen de acuerdo a los dos principales tipos de componentes con po-

tencia escalable:

1) **Memoria con potencia escalable**, por ejemplo el *Memory Management Infra-Structure for Energy Reduction* (Memory MISER) que redujo el consumo de memoria un 70% y el consumo de energía total un 30% (Tolentino *et al.*, 2007).

2) **Procesadores con potencia escalable**, como es la Calendarización con Prioridades y Escalado de Voltaje Dinámico (DVS, por sus siglas en inglés) fuera de línea, que obtuvo ahorro de energía de un 30% sacrificando 5% del desempeño del sistema (Ge *et al.*, 2005a).

En lo que se refiere al balanceo de cargas, se busca distribuir las cargas de trabajo a través de todo el sistema para obtener una utilización óptima de los recursos, minimizar el tiempo de respuesta y evitar sobrecargas del sistema. Por lo que esta estrategia se caracteriza como el punto de equilibrio entre consumo de energía y desempeño del sistema (Pinheiro *et al.*, 2001).

2.2.2 Manejo de energía en grids

La energía se ha convertido en un reto clave en sistemas distribuidos de gran escala, como lo son los Grids. Algunos trabajos se enfocan en la eficiencia de componentes específicos, tales como tarjetas de interfaz de red (Gunaratne *et al.*, 2005), discos de almacenaje (Allalouf *et al.*, 2009) y CPUs (Snowdon *et al.*, 2005). Existen modelos que se basan el la actividad del CPU (Fan *et al.*, 2007), otros basados en contadores de eventos (Merkel y Bellosa, 2006) y otros que se orientan al manejo de temperaturas (Fan *et al.*, 2007), balanceo de cargas (Merkel y Bellosa, 2006), calendarización de tareas y prendido y apagado de componentes (Chase *et al.*, 2001; Develder *et al.*, 2008).

Para el modelado de este tipo de sistemas se han hecho suposiciones que han resultado ser ciertas, como que el consumo de nodos homogéneos es el mismo, cuando éste depende de factores como acceso al disco, uso intensivo de comunicación a través de interfaces de red o procesamiento intensivo en el CPU. De hecho, hasta la ubicación en en el estante afecta el consumo de energía, debido a la influencia de las computadoras circundantes (Orgerie *et al.*, 2010).

Las estrategias para el manejo de energía son difíciles de elaborar para los sistemas distribuidos de gran escala debido a la falta de datos sobre el consumo energético de los componentes. Se han hecho modelos que suponen que el CPU es el componente que más energía consume (Fan *et al.*, 2007), pero debido a lo avances con DVS éste ya no es el caso, como se muestra en la Figura 4 (Minas y Ellison, 2009; Beloglazov *et al.*, 2010). De cualquier manera, se puede modelar el consumo de energía de acuerdo con la actividad del procesador (Fan *et al.*, 2007), aunque en los casos reales, la relación carga consumo no sea lineal (Orgerie *et al.*, 2010).

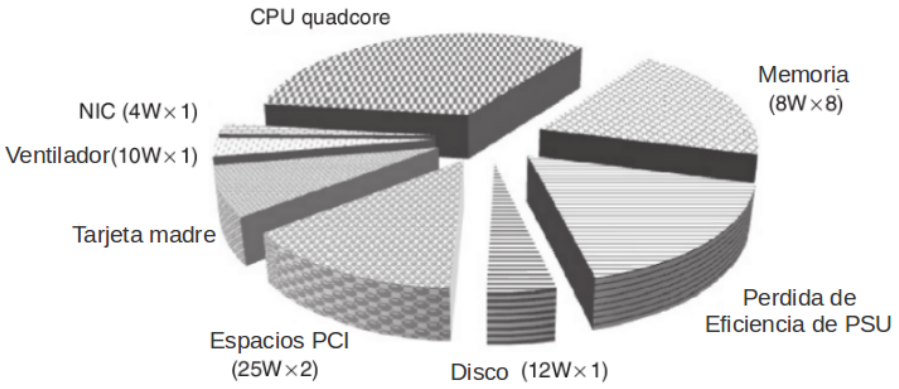

Figura 4. Consumo de energía de un servidor publicado adaptado de Beloglazov et al., 2010, p. 54.

También se debe considerar el gasto energético para mantener activa la presencia de las redes, esto se hace a través de técnicas de servidores proxy (Costa *et al.*, 2009;

Jimeno y Christensen, 2008). Aunque el gasto de energía no es linealmente proporcional al uso de banda ancha (Hlavacs *et al.*, 2009).

Así mismo, la relación entre uso de memoria y consumo de energía tampoco es lineal con el número de E/S por segundo. Ésto se debe a que las búsquedas se auxilian de un arreglo ordenado de llamadas de entradas y salidas, el cual reduce el consumo (Orgerie *et al.*, 2010).

Se ha demostrado que existen Grids que están subutilizados, que por lo tanto están gastando más recursos de lo necesario. En un estudio realizado en el Grid5000 (Cappello *et al.*, 2005), se muestra que los sitios de procesamiento trabajan a un 50.57% de su capacidad real (Orgerie *et al.*, 2008).

Apagar máquinas cuando no se estén usando y prenderlas sólo cuando se van a usar es una de las primeras estrategias estudiadas para el manejo de energía. El problema con estas estrategias ocurre cuando el prender y apagar ocurre intensivamente, por lo que se recomienda dejar un periodo ocioso para evitar apagar y prender la máquina en caso de llegar otra tarea (Chase *et al.*, 2001; Develder *et al.*, 2008), o usar un sistema de predicción de arribo de tareas como el de *Energy-Aware Reservation Infrastructure* (EARI) (Orgerie *et al.*, 2008). Ya que el prender una máquina crea picos de consumo, como se muestra en la Figura 5, y aunque no se considera en algunos modelos (Develder *et al.*, 2008), con que una máquina esté conectada, aunque esté apagada consume energía, el correspondiente a un 10% comparado con el estado ocioso (Orgerie *et al.*, 2010).

Existen modelos más sencillos que emulan de forma efectiva, para pruebas de algoritmos de calendarización, las características de grids reales (por ejemplo, EGEE/LCG Grid) (Develder *et al.*, 2008). Este modelo se presenta como la ecuación 2

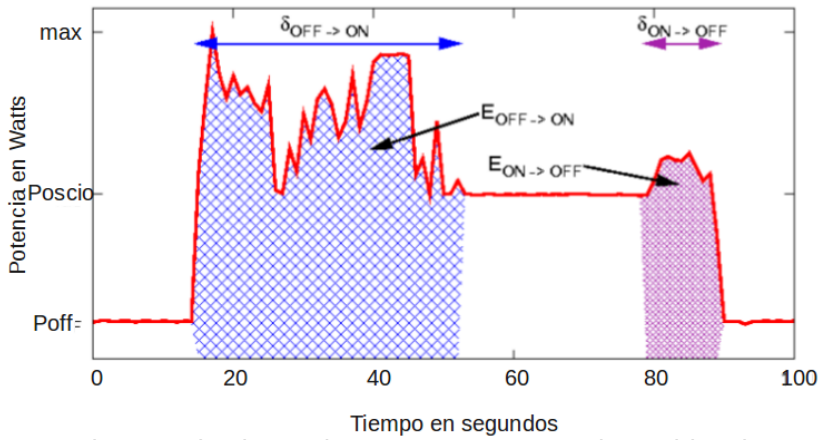

Figura 5. Consumo de energía al prender y apagar un servidor publicado en Orgerie et al., 2008, p. 173.

$$E_{serv} = E_{start} + E_{op} \text{ con } \left\{ \begin{array}{c} E_{start} = n_{start} \cdot P_{start} \cdot T_{start} \\ E_{op} = \int_t P_{op}(t)dt, \end{array} \right\} \tag{2}$$

donde E_{start} representa la energía requerida para prender un servidor, siendo n_{start} el número de veces que se prende, P_{start} la potencia requerida para prenderlo y T_{start} el tiempo requerido para prenderlo. De similar manera, E_{op} representa la energía usada para operar el grid, la cual depende de la potencia requerida en cada instante de tiempo $P_{op}(t)$ durante todo el tiempo t, como se muestra en la ecuación 3,

$$P_{op}(t) = P_{idle} + \sum_{c=1}^{C} \rho_c(t) \cdot P_{core} + BW_{serv}t + P_{newtw}, \tag{3}$$

donde $\rho_c(t) \in [1, 0]$ dependiendo si el núcleo está trabajando en el tiempo t, consume P_{core} y se le suma tanto el consumo en ocio P_{idle} y el consumo de energía causado por la red $BW_{serv}t + P_{newtw}$. Tomando los valores que se presentan en la Tabla 1.

Tabla 1. Valores del modelo de energía propuesto por Develder et al., 2008.

2	Valor	Descripción
P_{idle}	183.26 W	Potencia requerida en ocio
P_{core}	19.53 W	Potencia extra requerida cuando el núcleo trabaja
P_{start}	201.20 W	Potencia requerida para prender un servidor
T_{start}	89 s	Tiempo requerido para prender un servidor

2.2.3 Manejo de energía en grids de computadoras personales

Los grids de computadoras personales (también conocidos como grids oportunísticos) son un caso particular de los grids computacionales. Los recursos que componen este tipo de grids son extremadamente heterogéneos, tanto en poder de procesamiento, como en su respectivo consumo de energía (Ponciano y Brasileiro, 2010). Este tipo de grid está diseñado para ejecutar aplicaciones paralelas que pueden ser divididas en un gran número de tareas independientes, también llamadas BoTs, las cuales pueden ser usadas para un gran número de aplicaciones (Silva *et al.*, 2003; Anglano y Canonico, 2005; Cirne *et al.*, 2006; Anglano y Canonico, 2008; Ponciano y Brasileiro, 2010; Barrondo *et al.*, 2012). La eficiencia de las BoTs se debe al simple mapeo de una cola de trabajos a los recursos disponibles (Silva *et al.*, 2003).

Debido a la pequeña diferencia en el consumo de cuando una computadora está en ocio a cuando está trabajando, la computación realizada en estos grids, que explotan los ciclos de ocio de las computadoras que los componen, son eficientes en términos de eficiencia energética. Ésto se debe a que el usar esta capacidad de proceso "paga" el hecho de estar consumiendo la potencia requerida para mantener la máquina en ocio (Ponciano y Brasileiro, 2010). Aunque en el caso de los P2P-DG se demostró que

raramente están usados a su máxima capacidad (Iosup *et al.*, 2006). Por lo que la investigación del manejo de energía en estos grids se ha vuelto de interés comercial.

La calendarización consciente del consumo de energía en los grids puede brindar una notable disminución en el consumo de energía, que a su vez se extiende a ahorro económico, reducción de calor y un menor impacto en el ambiente. Hay estrategias para el manejo de energía basada en predicciones estáticas de la posible carga de trabajo generada por las tareas a ejecutar y los posibles requerimientos de hardware para ejecutarlas, y de esta forma regular el consumo de energía usando escalado dinámico de voltaje (DVS, por sus siglas en inglés), pero que a su vez varía la potencia de los componentes. Este algoritmo muestra una notable mejora en el consumo de energía, de 12-45% dependiendo en la configuración del grid, sin una notable degradación en desempeño, pero se comenta la necesidad de mejora del algoritmo de predicción (Sharma y Aggarwal, 2009).

Existen otras estrategias para el manejo de energía, como lo es el manejo de estados de energía predeterminados (Delaluz *et al.*, 2002), el manejo de modos de bajo consumo en las tarjetas de red (Krashinsky y Balakrishnan, 2002) y el apagado total de máquinas según su utilización con un intervalo de tiempo de gracia antes de hacer el cambio de estado (Ponciano y Brasileiro, 2010; Barrondo *et al.*, 2012).

Para poder aplicar esta última, se debe suponer que los agentes de software que manejan el grid tienen capacidades de *Wake-on-LAN*, lo que les permite hacer las transiciones de estado. Sin embargo, las heurísticas de calendarización varían, por ejemplo, se sabe que el algoritmo de Cola de Tareas con Replicación (WQR, por sus siglas en inglés) desperdicia ciclos al usar replicación (Silva *et al.*, 2003), la cual se traduce directamente a consumo de energía (Barrondo *et al.*, 2012). El utilizar metadatos como la eficiencia energética, que depende de la velocidad y el consumo de energía

($\text{eff}_i = \frac{s_i}{e_i}$), como heurística para hacer una calendarización eficiente ha demostrado tener buenos resultados (Ponciano y Brasileiro, 2010). Sin embargo, en casos reales esta información es difícil de obtener (Silva *et al.*, 2003; Orgerie *et al.*, 2010; Barrondo *et al.*, 2012).

2.2.4 Manejo de energía en nubes computacionales

El paradigma de cómputo en nube se refiere tanto al software como al hardware ofrecidos como servicios a través de Internet. Este paradigma ha hecho realidad el viejo sueño de ofrecer el cómputo como servicio, aparentando a través de la virtualización, recursos inagotables que sólo se usan y se pagan bajo demanda (Armbrust *et al.*, 2009). Pero estos "recursos inagotables" virtuales necesitan ser mapeados a recursos reales para tener sustento físico, los cuales pueden ser centros de datos (Weiss, 2007; Armbrust *et al.*, 2009) o grids (Murphy *et al.*, 2010).

La virtualización ha sido presentada como la panacea universal para la reducción del consumo de energía en los sistemas distribuidos de gran estala (Srikantaiah *et al.*, 2008), sin embargo, el consumo energético es el mismo con y sin virtualización (Orgerie *et al.*, 2010), aparte que por el hecho de tener un monitor de máquinas virtuales en ejecución constante, se podría alegar que el consumo es ligeramente mayor (Torres *et al.*, 2008).

Por lo que el ahorro de energía en éste paradigma depende totalmente de las estrategias de ahorro de energía de los recursos que sustentan los servicios. Estas estrategias son las mismas que se usan para Clusters y Grids Computacionales, pero con la diferencia que deben de cumplir los requerimientos de disponibilidad, confiabilidad y desempeño de un acuerdo de nivel de servicio (SLA, por sus siglas en inglés) (Berral *et al.*, 2010).

El mecanismo mayormente usado en éstos centros de datos es DCD, ya que entre

menos hardware se requiera, se requiere menos energía. Este tipo de estrategia requiere de información de las futuras cargas de trabajo, la cual no siempre se tiene. Por lo que se han desarrollado algoritmos que tratan de predecir la futura carga para poder decidir si apagar o no cual máquina (Berral *et al.*, 2010; Orgerie *et al.*, 2010).

2.3 Calendarización

En el corazón de todo sistema distribuido, un calendarizador controla el como los programas se ejecutan en los recursos disponibles. Debido al tamaño y la naturaleza dinámica de los grids, el proceso de seleccionar y asignar tareas computacionales a recursos disponibles en el grid debe hacerse de forma automática y eficiente (Schwiegelshohn *et al.*, 2008; Ramírez-Alcaraz *et al.*, 2011; Quezada-Pina *et al.*, 2012). Este calendarizador actúa como proxy para docenas, cientos o miles nodos computacionales, encargándose del descubrimiento y selección de recursos y de la ejecución de las tareas computacionales (Chee y Franklin, 2010). En estos sistemas, debido a que las aplicaciones comparten recursos (procesamiento, memoria, ancho de banda, etc.), para obtener un buen desempeño se debe hacer un calendario con la selección del número y tipo de recursos usados en que aplicación basado en las características de ésta, así como la ubicación y volumen de los datos de entrada y salida (Foster y Kesselman, 1999).

Aunque los grids de computadoras personales estén restringidos a aplicaciones BoT, existen la tendencia para calendarizar flujos de trabajo cada ves más complejos en grids tradicionales. Pero una calendarización de flujos de trabajo efectiva requiere de una eficiente asignación de tareas a recursos limitados, la cual es actualmente sujeto de investigación en muchos trabajos (Hirales-Carbajal *et al.*, 2012).

Siendo T un conjunto de n tareas y m el número de procesadores en un grid G, un

calendario S de T en G se define como el conjunto finito de tripletas (v, p, t), donde $v \in T$, $p|1 \leq p \leq m$ y t es el tiempo de inicio de ejecución. Además para cada $v \in T$ existe al menos un $(v, p, t) \in S$ y no existe $(v, p, t), (v', p, t') \in S$ tal que $t \leq t' \leq t + d$ donde $t + d$ es el tiempo de terminación de v (Fujimoto y Hagihara, 2003).

2.3.1 Resultados previos

Se ha demostrado que el problema de calendarización en grid es más complejo que el correspondiente problema de calendarización en múltiples procesadores, y que algoritmos de calendarización en lista conocidos, no pueden garantizar una razón de competitividad del tiempo de ejecución constante (Schwiegelshohn *et al.*, 2008). El problema de calendarizar se describe por tres campos $(\alpha|\beta|\gamma)$ donde | es un delimitador; α describe el sistema de ejecución, por ejemplo P_m indica que los m procesadores son idénticos, P indica que los m procesadores son idénticos y m varía, Q_m indica que los m procesadores tienen diferentes razones de velocidad enteras, Q indica que los m procesadores tienen diferentes razones de velocidad enteras y m varía; β describe la carga de trabajo, por ejemplo $p_j = 1$ indica tiempos de procesamiento unitarios, o "p_j impredecible" se refiere al caso no clarividente, donde no se sabe p_j; γ, por su parte, describe el criterio de la función objetivo, como puede ser C_{max} (Fujimoto y Hagihara, 2003). Siguiendo la misma notación se presenta la Tabla 2.

2.3.2 Calendarización en grids de computadoras personales

Calendarizar aplicaciones en un ambiente tan heterogéneo, dinámico y altamente distribuido como lo es un grid de computadoras personales no es una tarea trivial. Debido a estas características se puede alegar que las aplicaciones de bolsas de tareas (BoT, por sus siglas en inglés), que están compuestas por un conjunto de tareas completamente

independientes, son las aplicaciones más apropiadas para los ambientes de grid actuales (Silva *et al.*, 2003). Existen muchas aplicaciones BoT, minería de datos, simulaciones Monte Carlo, cálculos fractales, manipulación de imágenes, etc.

De cualquier manera, por más sencillo que parezca la calendarización de aplicaciones BoT, que son realmente simples, por la naturaleza heterogénea y dinámica del grid esta tarea se complica. Más aún, es sabido que para realizar un buen calendario es necesario tener información del grid y de las tareas. Sin embargo, el obtener esta información muchas veces no es posible debido a la amplia distribución del grid. Se han hecho proyectos para el monitoréo de grids (Lowekamp *et al.*, 1999), pero la implementación a gran escala presenta muchos obstáculos. De la misma manera, los resultados que usan estimados de usuario han demostrado ser inexactos y dando como resultado utilización ineficiente de las máquinas (Hirales-Carbajal *et al.*, 2012). Aún así, la gran mayoría de algoritmos de calendarización suponen que tienen acceso a información completa y confiable.

Al usar un algoritmo con replicación de tareas (Workqueue with Replication, WQR) se obtiene un desempeño comparable a la calendarización que utiliza información (Silva *et al.*, 2003). Además se puede agregar tolerancia a fallas al algoritmo utilizando reinicio automático de tareas fallidas, y puntos de control para el aprovechamiento de trabajo ya realizado (Workqueue with Replication-Fault Tolerant, WQR-FT) (Anglano y Canonico, 2005). Por último, se introduce la calendarización de múltiples aplicaciones BoT, utilizando un algoritmo de dos capas, la superior encargada de la selección de la aplicación por medio de alguna política (FIFO, Round Robin, etc.) y la inferior encargada de la calendarización de las tareas usando WQR-FT (Anglano y Canonico, 2008). Aunque estos trabajos han brindado una buena base, hace falta una mayor investigación en algoritmos con bajos requerimientos de información, que debido a su

simplicidad, son los más fáciles de implementar en un ambiente de grid real.

Tabla 2. Resultados previos en calendarización

Problema	Resultado
$P\|\|C_{max}$	NP-Duro (Garey y Johnson, 1978)
$P_2\|\|C_{max}$	NP-Difícil (Lenstra *et al.*, 1977)
$P\|\|C_{max}$	Es aproximable con el factor $\frac{4}{3} - \frac{1}{3m}$ (Graham, 1969)
$P\|\|C_{max}$	Es aproximable con el factor $(1+\epsilon)$ en tiempo $O((\frac{n}{\epsilon})^{\frac{1}{\epsilon^2}})$ para todo $\epsilon > 0$ (Hochbaum y Shmoys, 1987)
$P\|\|C_{max}$	Es aproximable con el factor $\frac{6}{5} + 2^{-k}$ en tiempo $O(n(k + logn))$ donde k es cualquier entero posible (Hochbaum y Shmoys, 1987)
$P\|\|C_{max}$	Es aproximable con el factor $\frac{7}{6} + 2^{-k}$ en tiempo $O(n(km^4 + logn))$ donde k es cualquier entero posible (Hochbaum y Shmoys, 1987)
$P_m\|\|C_{max}$	Es aproximable con el factor $(1+\epsilon)$ en tiempo $O(\frac{n}{\epsilon})$ para todo $\epsilon > 0$ (Horowitz y Sahni, 1976)
$P\|p_j = 1\|C_{max}$	Es trivial; el tiempo de terminación 1 es $\lceil \frac{n}{m} \rceil$
$P\|p_j$ impredecible $\|C_{max}$	Es aproximable con un factor $(2 - \frac{1}{m})$ (Graham, 1966)
$P\|p_j$ impredecible $\|C_{max}$	Es aproximable con un factor $(2 - \frac{1}{m} - \epsilon_m)$ donde ϵ_m es un positivo real dependiente de m (Galambos y Woeginger, 1993)
$Q\|p_j = 1\|C_{max}$	Puede ser resuelto en tiempo $O(n^2)$ (Graham *et al.*, 1979)
$Q_2\|\|C_{max}$	Es aproximable con el factor $(1+\epsilon)$ en tiempo $O((\log m + \log(\frac{3}{\epsilon}))(n + \frac{8}{\epsilon^2})m \cdot n^{3+\frac{40}{\epsilon^2}})$ para todo $0 < \epsilon \leq 1$ (Hochbaum y Shmoys, 1988)
$Q; s_{jk}\|\|C_{max}$	Es aproximable con el factor $1 + \frac{m(\log_e(m-1)+1)}{n}$ del óptimo (Fujimoto y Hagihara, 2003)

Capítulo 3

Metodoloía

Con el fin de completar los objetivos y contestar las preguntas de investigación, se deben diseñar distintos casos de estudio y modelos que permitan la evaluación de las distintas métricas para poder discernir la mejor estrategia.

3.1 Modelo de Grid de Computadoras Personales

"En la mayoría de los problemas de calendarización reales, los grandes números y otras restricciones casi siempre evitan que los estudios teóricos obtengan resultados significativos. Esto es particularmente cierto con los grids, los cuales están sujetos a heterogeneidad, comportamiento dinámico, muchos tipos de trabajos, y otras restricciones. Debido a esto, el modelo para cualquier estudio teórico de calendarización en grid debe ser una abstracción de la realidad" - Schwiegelshohn *et al.* (2008), pp. 1061-1062.

Mientras que las máquinas en grids reales normalmente tienen distintos tipos de heterogeneidad, como diferente hardware, sistema operativo y software, los avances en virtualización hacen menos relevante estas diferencias en lo que concierne a la administración de recursos (Tchernykh *et al.*, 2010). Este trabajo se restringe a heterogeneidad en velocidad de proceso y eficiencia energética. Se usó un modelo que se enfoca en algunos de los aspectos más importantes de los Grids de Computadoras Personales. Este modelo consiste en un conjunto de m máquinas $G = \{M_1, M_2, ..., M_m\}$. Siendo s_i, e_i, y $\text{eff}_i = \frac{s_i}{e_i}$ la velocidad, el consumo energético y la eficiencia energética de la máquina M_i, respectivamente. Además, $S = \sum_{i=1}^{m} s_i$ es la velocidad total de G, $s_{\max}$ es la velocidad de procesamiento máxima del conjunto de velocidades $G_s = \{s_1, s_2, ..., s_m\}$ que son las

velocidades correspondientes a G.

Debido a que el número de máquinas y su disponibilidad cambia a través del tiempo, y que no se considera una cola de tareas local en cada máquina, una máquina M_i sólo puede trabajar cuando $M_i \in A(t)|A(t) \subset G \wedge w_i(t) = 0$, donde $A(t)$ es el conjunto de máquinas disponibles en el tiempo t y $w_i(t) = 0$ cuando la máquina M_i no está trabajando en tiempo t.

3.2 Modelo de Energía

En este trabajo el consumo de energía se mide en watts por hora (Wh) o kilo watts por hora (kWh). Con el fin de poder medir la cantidad de energía necesaria en un P2P-DG

$$E^{\text{grid}} = \sum_{i=1}^{m} (E_i^{\text{prendeMáquina}} + E_i^{\text{opMáquina}}), \tag{4}$$

se modificó un modelo de consumo para grids tradicionales (Develder *et al.*, 2008) y se adaptó para este tipo de grids (Barrondo *et al.*, 2012). En este modelo, el consumo de energía de cada máquina M_i de G se calcula considerando la energía necesaria para prenderla

$$\begin{aligned} E_i^{\text{prendeMáquina}} = &\left(n_i^{\text{prendeMáquina}} \cdot T_i^{\text{prendeMáquina}} \cdot P_i^{\text{prendeMáquina}}\right) + \\ &\left(n_i^{\text{inicioNúcleo}} \cdot T_i^{\text{inicioNúcleo}} \cdot P_i^{\text{inicioNúcleo}}\right), \end{aligned} \tag{5}$$

donde se considera cada una de las $n_i^{\text{prendeMáquina}}$ veces que se prendió M_i, además de estar operacional sólo después de $T_i^{\text{prendeMáquina}}$ tiempo, y requerir de $P_i^{\text{prendeMáquina}}$ de potencia para hacerlo. Similarmente, para cada una de las $n_i^{\text{inicioNúcleo}}$ veces que se prendió el núcleo de M_i, estuvo operacional sólo después de $T_i^{\text{inicioNúcleo}}$ tiempo, y requerir $P_i^{\text{inicioNúcleo}}$ de potencia para hacerlo.

Debido a este retraso (latencia) y el respectivo consumo de energía al prender una máquina, para poder hacer transiciones de estado efectivas y evitar demasiados cambios

de estado (prendido o apagado), sólo se apaga M_i cuando haya pasado suficiente tiempo de ocio $T_i^{\text{apagaMáquina}}$ como para compensar los gastos generales de la transición de estado en caso de que llegase otra tarea en ese tiempo. En sistemas reales, se tiene poca o ninguna información sobre tareas futuras.

La energía necesaria para operar cada máquina de G

$$E_i^{\text{opMáquina}} = \sum_{t=1ms}^{C_{max}} P_i^{\text{máquina}}(t), \tag{6}$$

se calcula usando la potencia requerida por M_i para operar en Δt, discretizando la función con $\Delta t = 1ms$

$$P_i^{\text{máquina}}(t) = o_i(t) \cdot (P_i^{\text{máquinaPrendida}}(t) + P_i^{\text{núcleo}}(t)), \tag{7}$$

y verificando $o_i(t)$ el cual es un valor booleano que denota si la máquina está prendida en tiempo t. Además, $P_i^{\text{máquina}}(t)$ (ecuación 7) consta de una parte constante $P_i^{\text{máquinaPrendida}}(t)$, que denota el requerimiento de potencia de la máquina M_i para mantenerse en estado ocioso, y una parte variable,

$$P_i^{\text{núcleo}}(t) = v_i(t) \cdot (P_i^{\text{núcleoPrendido}}(t) + w_i(t) \cdot P_i^{\text{núcleoTrabajo}}(t)), \tag{8}$$

que varía si M_i está trabajando.

$P_i^{\text{núcleo}}(t)$ (ecuación 8) es la potencia requerida por el núcleo de procesamiento de la máquina M_i en tiempo t. De igual manera consiste de una parte constante $P_i^{\text{núcleoPrendido}}(t)$, que corresponde a la potencia requerida por mantener prendido pero ocioso el núcleo, y una parte variable $P_i^{\text{núcleoTrabajo}}(t)$ que corresponde a la potencia extra requerida para utilizar el núcleo. Similarmente, $v_i(t)$ y $w_i(t)$ son valores booleanos que denotan si el núcleo está prendido o apagado y trabajando u ocio en tiempo t respectivamente.

Cuando un núcleo está apagado no requiere potencia, pero cuando está prendido requiere al menos $P_i^{\text{núcleoPrendido}}(t)$, aunque no esté trabajando. Adicionalmente, cuando un núcleo está trabajando, requiere $P_i^{\text{núcleoTrabajo}}(t)$ extra para funcionar.

Por lo tanto, el modelo supone que los requerimientos de potencia de una máquina son esencialmente constantes, sin importar la actividad de ésta. De esta forma, $P_i^{\text{máquinaPrendida}}(t)$ y $P_i^{\text{núcleoPrendido}}(t)$ incluyen la potencia requerida por los demás componentes de la máquina, como el enfriamiento, cuyos requerimientos no dependen de si ésta está trabajando o no.

3.3 Modelo de Carga de Trabajo

Se atacó un problema de calendarización fuera de línea, donde se deben calendarizar n tareas de una bolsa de tareas $BoT = \{T_1, T_2, ..., T_n\}$ en el conjunto G de m máquinas uniformes. Cada tarea T_j es descrita por p_j, que es el tiempo de ejecución en la máquina más lenta posible $(M_i|s_i = 1)$, el cual es una abstracción del número de operaciones de punto flotante de T_j, y para fines de calendarización se desconoce hasta haber terminado de ejecutar la tarea (caso no-clarividente).

Aunque los grids exhiben un comportamiento "en-línea", se sabe que las tareas permanecen en colas por una cantidad de tiempo significante, por lo que la calendarización fuera de línea es beneficiosa para la calendarización en-línea. Además varias estrategias de calendarización fuera de línea tienen buen desempeño en escenarios en-línea, y por la teoría se sabe que los limites de desempeño de las estrategias de calendarización fuera de línea pueden ser aproximadas para el caso en-línea (Shmoys *et al.*, 1995; Hirales-Carbajal *et al.*, 2012).

El número de operaciones de punto flotante total de la aplicación BoT está dado por

$p = \sum_{j=1}^{n} p_j$. Todas las tareas pertenecientes a la misma BoT tienen el mismo tiempo de liberación (r), y cualquier tarea T_j puede ser asignada a cualquier máquina que esté disponible en ese tiempo ($T_j \rightarrow M_i | M_i \in A(t)$). Los tiempos de inicio de ejecución y terminación de una tarea T_j en la máquina M_i se denotan como st_j y $c_j = st_j + \frac{p_j}{s_i}$ correspondientemente. El tiempo de terminación de la BoT está dado por el tiempo de terminación máximo $C_{max} = \max_{j=1,\ldots,n}(c_j)$.

3.4 Estrategias de Calendarización

Se parte del esquema básico de *Cola de Trabajos* (*WQ*, por sus siglas en inglés), donde el calendarizador asigna de manera aleatoria las tareas de la bolsa hasta que ésta está vacía. Además, este esquema básico se extiende en el algoritmo *Cola de trabajos con Replicación* (*WQRx*, por sus siglas en inglés) presentado en Silva *et al.* (2003), donde se agregan x réplicas a cada tarea como clave para sobrellevar una mala asignación estadísticamente inminente (por ejemplo, asignar la tarea más larga a la máquina más lenta), pero aumentando el número de tareas a calendarizar a $n \cdot (1 + x)$. Cuando no hay más tareas para asignar, una réplica aleatoriamente escogida es asignada a alguna máquina disponible (véase Figura 1), y cuando una máquina termina una tarea (o réplica), todas las réplicas de la tarea terminada son canceladas, lo que genera pérdida de ciclos y por lo tanto desperdicio de energía (Barrondo *et al.*, 2012).

3.4.1 Niveles de información para calendarización

Para mejorar el desempeño de los grids, en esta tesis se propone extender el esquema de *WQRx* propuesto en Silva *et al.* (2003), se añade niveles de información a los algoritmos, los cuales son usados para tomar decisiones mas acertadas. Esta información, en los grids reales se puede obtener usando programas de pruebas ("Benchmarks") como

Dhrystone, Whetstone y Malikstone (Awan y Jarvis, 2012).

Como se muestra en la Tabla 3, el nivel 0, es libre de información, y se evalúan *WQRx* y Cola de Trabajos con Historial *WQRx-h*, el cual basado en el Axioma 1, selecciona la máquina más rápida, o en caso de no tener información selecciona una máquina aleatoria.

Axioma 1 Si el tiempo de inicio de una réplica cancelada es menor o igual que el de la réplica terminada, entonces la máquina que terminó la tarea es más rápida que la máquina que canceló la réplica.

Tabla 3. Niveles de información de algoritmos basados en *WQR*.

Estrategia	Nivel	Descripción
WQR#	0	Aleatorio (Silva *et al.*, 2003)
WQR#_h	0	Máxima Velocidad ($M_i \mid \max(A_{\mathrm{s}}(t))$) o Aleatorio
WQR#_s	1	Máxima Velocidad ($M_i \mid \max(A_{\mathrm{s}}(t))$)
WQR#_eff	1	Máxima Eficiencia ($M_i \mid \max(A_{\mathrm{eff}}(t))$)
WQR#_seff	1	Máxima Velocidad & Eficiencia ($M_i \mid \max(A_{\mathrm{seff}}(t))$)

Los algoritmos del nivel 1 sólo utilizan información de de los recursos de cómputo. Los algoritmos evaluados en este nivel son: Cola de trabajos con Velocidad *WQRx-s*, Cola de trabajos con Eficiencia Energética *WQRx-s* y Cola de trabajos con Velocidad y Eficiencia Energética *WQRx-seff*. Los tres algoritmos asignan según la heurística correspondiente: máxima Velocidad ($M_i \mid \max(A_{\mathrm{s}}(t))$), máxima Eficiencia ($M_i \mid \max(A_{\mathrm{eff}}(t))$) y máxima Velocidad & Eficiencia ($M_i \mid \max(A_{\mathrm{seff}}(t))$). Donde $A_s(t) = \{s_1, s_2, ..., s_m\}$, $A_{\mathrm{eff}}(t) = \{\mathrm{eff}_1, \mathrm{eff}_2, ..., \mathrm{eff}_m\}$ y $A_{\mathrm{seff}}(t) = \{s_1 \cdot \mathrm{eff}_1, s_2 \cdot \mathrm{eff}_2, ..., s_m \cdot \mathrm{eff}_m\}$ son los conjuntos de velocidades, eficiencias energética y velocidades con eficiencias energética respecto a

las máquinas que se encuentren en el conjunto de máquinas disponibles $A(t)$.

3.5 Método de Evaluación

Aunque la mayoría de las estrategias de evaluación consideran un solo criterio, el escenario de grid contiene aspectos que son multi-objetivo por naturaleza. Por ejemplo, el problema de la calidad de servicio (QoS, por sus siglas en inglés) es representativo de las necesidades del usuario, por otra parte, los proveedores de servicio pueden tener necesidades en conflicto con la de los usuarios, como puede ser optimizar la utilización de recursos (Hirales-Carbajal *et al.*, 2012). En este caso, se evalúan dos criterios, un factor de aproximación (ρ), que representa la QoS y el consumo de energía que se presenta en la ecuación 4 (E^{grid}), que es el factor de optimización de recursos a considerar. Debido a que es un problema multi-criterio, un buen algoritmo de calendarización debe minimizar el factor de aproximación (buena QoS) y al mismo tiempo minimizar el consumo de energía.

El usar el tiempo de ejecución, o *makespan*, como objetivo de minimización tiene sus desventajas (Fujimoto y Hagihara, 2003), en especial en los escenarios en-línea con tareas independientes, pero al ser fácil de manejar es frecuentemente usado. Normalmente en calendarización se evalúan los algoritmos determinando las cotas de los criterios de competitividad (Schwiegelshohn *et al.*, 2008). En este caso,

$$\rho = \frac{C_{max}}{\bar{C}^{opt}_{max}}, \tag{9}$$

depende de una cota inferior del tiempo de ejecución $\bar{C}^{opt}_{max}$ y del tiempo de ejecución C_{max}. Debido a que no es viable determinar el tiempo de ejecución óptimo C^{opt}_{max}, se utiliza un limite inferior teórico

$$C^{opt}_{max} \geq \bar{C}^{opt}_{max} = \max\left\{\frac{max_j(p_j)}{s_{max}}, \frac{\sum_{j=1}^{n} = p_j}{S}\right\}, \tag{10}$$

en vez del tiempo de ejecución óptimo.

3.5.1 Degradación

El problema se puede simplificar a un sólo objetivo a través de diferentes métodos de combinación de objetivos. Existen también varias formas de modelar preferencias, por ejemplo, se les puede dar un criterio específico o asignarles importancia relativa a cada criterio. Ésto se hace definiendo un peso para cada criterio o clasificando y ordenándolas según su importancia.

Para proveer una guía efectiva para escoger la mejor estrategia, se realizó un análisis conjunto de ambas métricas de acuerdo con la metodología propuesta en Tsafrir *et al.* (2007), y usada para el problema de calendarización en grids en Ramírez-Alcaraz *et al.* (2011); Barrondo *et al.* (2012). Ellos introducen una solución que supone la misma importancia para cada métrica. El objetivo es encontrar una estrategia que tenga buen desempeño en todos los casos de estudio, con la expectativa de que tenga buen desempeño en otras condiciones, por ejemplo, con diferentes configuraciones de grid y cargas de trabajo

El análisis se hace de la siguiente forma. Primero, se evalúa el porcentaje de degradación en desempeño de cada estrategia bajo cada métrica. Ésto se hace relativo a la estrategia con mejor desempeño por caso de estudio de la siguiente manera: $\left(\frac{\text{valor de la métrica}}{\text{mejor valor de la métrica}} - 1\right) \cdot 100$. Posteriormente, se promedian estos valores y se clasifican las estrategias.

La mejor estrategia, la que tenga la menor degradación, se clasifica como 1. Nótese que se trata de identificar estrategias que tengan un buen y confiable desempeño en diferentes escenarios, por lo que se trata de encontrar un compromiso que considere todos los casos de estudio. Por ejemplo, la clasificación de estrategias en promedio no

puede ser la misma para cada una de las métricas individualmente o para cualquier caso de estudio en particular.

Se presentan los promedios de degradación de los valores de las métricas para evaluar el desempeño de las estrategias y mostrar si alguna de las estrategias tiende a dominar. El enfoque de la degradación provee un porcentaje de mejora, pero no muestra intuitivamente los efectos físicos de las métricas en cuestión.

3.5.2 Perfiles de desempeño

Para poder analizar estos posibles efectos negativos de permitir que pequeñas porciones de las instancias del problema con una alta desviación dominen las conclusiones que están basadas en promedios, y para ayudar interpretar los datos generados por el proceso de evaluación, se presentan los perfiles de desempeño de las estrategias.

El perfil de desempeño $\rho(\tau)$ es una función constante, discreta y no-decreciente que presenta la razón de desempeño $r = \frac{\text{valor de la métrica}}{\text{mejor valor de la métrica}}$ para un factor τ (Dolan y Moré, 2002; Dolan *et al.*, 2006)

El perfil de desempeño de cada métrica se calcula con la ecuación

$$\rho_{metric}(\tau) = \frac{\text{tamaño}\{resultados | resultados \leq \tau\}}{n_{\text{exp}}}, \tag{11}$$

y después se promedia con la ecuación

$$\bar{\rho}_{metric}(\tau) = \frac{\sum^{n_{metrics}} \rho_{metric}(\tau)}{n_{metrics}}. \tag{12}$$

Donde tamaño$\{resultados | resultados \leq \tau\}$ es el número de experimentos que tienen un valor menor o igual a τ y n_{exp} es el numero de experimentos total. La mejor estrategia debiera tener una alta probabilidad de obtener una baja degradación.

3.5.3 Costo de desempeño

Como otra forma para encontrar la relación entre consumo de energía y calidad de servicio, además de convertir el problema bi-objetivo a mono-objetivo y tener un mayor entendimiento de los resultados, se propone en este trabajo el objetivo de *costo de desempeño* dada por

$$SC = \frac{E^{\text{grid}}}{D}, \tag{13}$$

el cual combina los valores de ambas métricas.Obteniendo la cantidad de energía requerida para alcanzar cierto desempeño

$$D = \frac{\sum_{j=1}^{n} p_j}{C_{max}}, \tag{14}$$

en cada escenario. De esta forma D representa una abstracción el número de operaciones de punto flotante por segundo o desempeño del calendario. Este objetivo a minimizar, que busca obtener el menor consumo de energía con la mayor velocidad posible, da una perspectiva más práctica a los resultados obtenidos.

Capítulo 4

Experimentos WQR

La primera etapa del estudio consistió en replicar el trabajo expuesto por Silva *et al.* (2003), y los resultados fueron presentados en Barrondo *et al.* (2012). Todos los experimentos fueron realizados utilizando el Framework de Simulación de Grid Teikoku (tGSF, por sus siglas en inglés). tGSF es un simulador diseñado para rastros de cargas estándar que es usado para estudiar problemas de manejo de recursos en Grid (Hirales-Carbajal *et al.*, 2010). Se modificó tGSF para poder incluir P2P-DGs, modelos de energía y replicación. Se realizó un análisis de estrategias de calendarización en grids con heterogeneidad de recursos, tareas y granularidad de BoTs.

4.1 Diseño Experimental

Se deben recalcar dos aspectos importantes para la evaluación de desempeño por una parte, cargas de trabajos representativas son necesarias para obtener resultados confiables. Por otra parte, es importante tener un buen ambiente de pruebas para poder obtener resultados reproducibles y comparables. Nótese, que al fijar $S = \sum_{i=1}^{m} s_i$ y $p = \sum_{j=1}^{n} p_j$, las diferencias en los resultados dependen exclusivamente del algoritmo de calendarización.

4.1.1 Grids

Se consideran cinco escenarios de grid para la evaluación. Al igual que en Silva *et al.* (2003), los grids se diferencian por la heterogeneidad en su velocidad de procesamiento,

la cual es dada por la distribución uniforme

$$s_i = U\left(10 - \frac{hm}{2}, 10 + \frac{hm}{2}\right), \tag{15}$$

se fijó $S = \sum_{i=1}^{m} s_i = 1000$ para todos los grids y para generar los cinco grids se usó $hm = \{1, 2, 4, 8, 16\}$ obteniendo los valores que se muestran en la Figura 6 y en el Apéndice A.1.

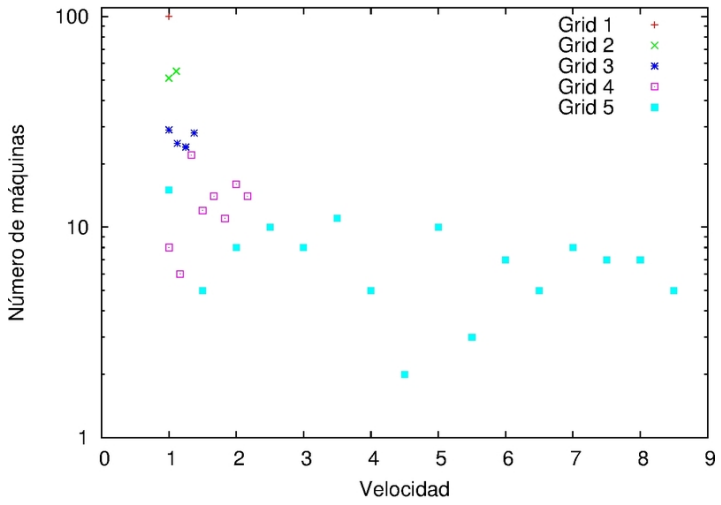

Figura 6. Histograma de velocidades de los distintos grids.

La configuración de valores del modelo de energía se adaptó de la propuesta en Develder *et al.* (2008). Esta configuración se describe en la Tabla 4.

4.1.2 Cargas de trabajo

Se consideran veinte aplicaciones como carga de trabajo. Éstas se dividen en cuatro grupos y cinco subgrupos que se caracterizan por la diferencia en el número de operaciones de punto flotante promedio ($\bar{p_j}$) de cada BoT. Los diferentes grupos se definen por $\bar{p_j} = \{1000, 5000, 25000, 125000\}$, lo que hace que varíe el número de tareas por BoT, como se muestra en la Figura 27. Los subgrupos corresponden a la variación de

Tabla 4. Valores del modelo de energía propuesto por Develder et al., 2008.

Notación	Valor	Relativos	Descripción
$P_i^{\text{núcleoTrabajo}}(t)$	19.53 W	0.1	Potencia extra requerida cuando el núcleo trabaja
$P_i^{\text{núcleoPrendido}}(t)$	8.26 W	0.04	Potencia requerida para mantener un núcleo en ocio
$P_i^{\text{máquinaPrendida}}(t)$	175.0 W	0.86	Potencia requerida para mantener un núcleo en ocio
$P_i^{\text{prendeMáquina}}$	201.20 W	0.99	Potencia requerida para prender una máquina
$T_i^{\text{prendeMáquina}}$	89 s		Tiempo requerido para prender una máquina
$T_i^{\text{apagaMáquina}}$	10 s		Tiempo de espera para apagar una máquina

$\bar{p_j}$ un 0%, 25%, 50%, 75% y 100%, en una distribución uniforme, como en Silva *et al.* (2003). En todas las aplicaciones se fija $p = \sum_{j=1}^{n} p_j = 3600000$. Por lo tanto, se presentan diferentes escenarios de Grid con diferentes cargas de trabajo que varían de 36 a 0.29 tareas por máquina, como se nota en la Tabla 5.

4.2 Resultados

Se confirmó que el uso de replicación sirve para reducir el tiempo de procesamiento, lo que es equivalente a mejorar la QoS. De la misma manera, se encontró que en promedio, esta estrategia de mejora de QoS tiene mayor impacto cuando el número de operaciones

Tabla 5. Relación entre tiempo de ejecución y número de tareas por máquina.

$\bar{p_j}$	n	Número de tareas por máquina
1000	3600	36
5000	720	7.2
25000	144	1.44
125000	29	0.29

de punto flotante promedio de las tareas es mayor, mientras que con las tareas con menor número de operaciones, no hay una diferencia entre el uso y no uso de réplicas que justifique el uso de las mismas, como se muestra en la Figura 7 y la Tabla 6.

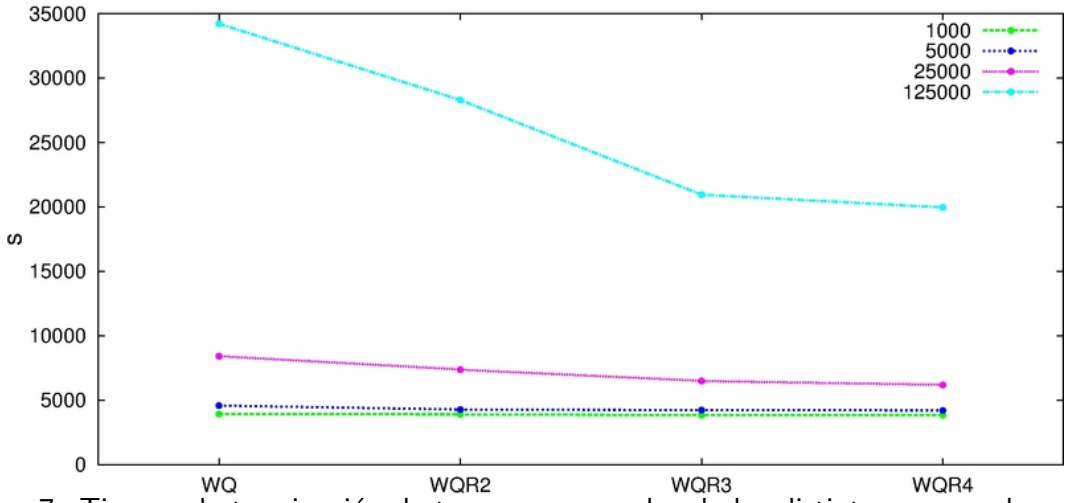

Figura 7. Tiempo de terminación de tareas en segundos de los distintos grupos de aplicaciones $\bar{p_j} = \{1000, 5000, 25000, 125000\}$ por cada umbral de replicación.

Aunque trivial, los resultados muestran que entre más réplicas se liberen, se consume más energía. Siendo esto más notorio al aumentar el número de operaciones de punto flotante de las tareas. Ésto se debe a que las réplicas están más tiempo en ejecución antes de que alguna de ellas termine y las demás sean canceladas, dando por perdidos los ciclos de procesamiento utilizados y por ende la energía requerida para este procesamiento,

Tabla 6. Tiempo de terminación de las tareas en segundos de los distintos grupos de aplicaciones $\bar{p_j} = \{1000, 5000, 25000, 125000\}$ por cada umbral de replicación.

Réplicas	$\bar{p_j} = 1000$	$\bar{p_j} = 5000$	$\bar{p_j} = 25000$	$\bar{p_j} = 125000$
0	3928.8	4584.84	8421.52	34230.56
1	3898.92	4272.68	7364.16	28300.56
2	3844.84	4231.76	6489.44	20961
3	3839.8	4202	6178.72	19970.76

como se puede ver en la Figura 8 y en la Tabla 7.

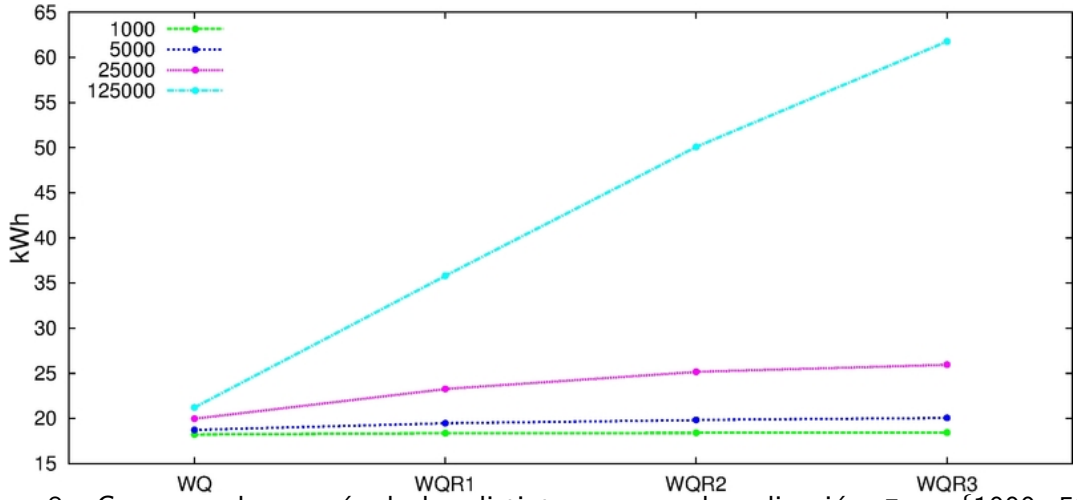

Figura 8. Consumo de energía de los distintos grupos de aplicación $\bar{p_j} = \{1000, 5000, 25000, 125000\}$ por cada umbral de replicación.

4.2.1 Degradación

Como se muestra en la Figura 9 y la Tabla 8, al promediar los porcentajes de degradación de los valores de las métricas en todos los grids, se encuentra que el aumento en el consumo de energía tiene mayor impacto que la reducción en el factor de aproximación (mejor QoS). Por lo que en promedio en todos los grids y ambas métricas teniendo la

Tabla 7. Consumo de energía (kWh) de los distintos grupos de aplicación $\bar{p_j} = \{1000, 5000, 25000, 125000\}$ por cada umbral de replicación.

Réplicas	$\bar{p_j} = 1000$	$\bar{p_j} = 5000$	$\bar{p_j} = 25000$	$\bar{p_j} = 125000$
0	18.22	18.73	19.97	21.21
1	18.37	19.48	23.26	35.81
2	18.41	19.84	25.17	50.08
3	18.44	20.07	25.96	61.77

misma importancia, el algoritmo con mejor desempeño es WQ, el que no tiene replicas.

Sin embargo, una de las características de este tipo de análisis es que se puede variar la importancia de las métricas. Como un ejemplo de ésto y con el fin de encontrar un compromiso entre QoS y E^{grid} se agregó peso o importancia a ρ. Con esta modificación al promedio se encontró que con dos réplicas se obtiene el mejor desempeño, siempre y cuando la QoS sea más importante que el consumo de energía.

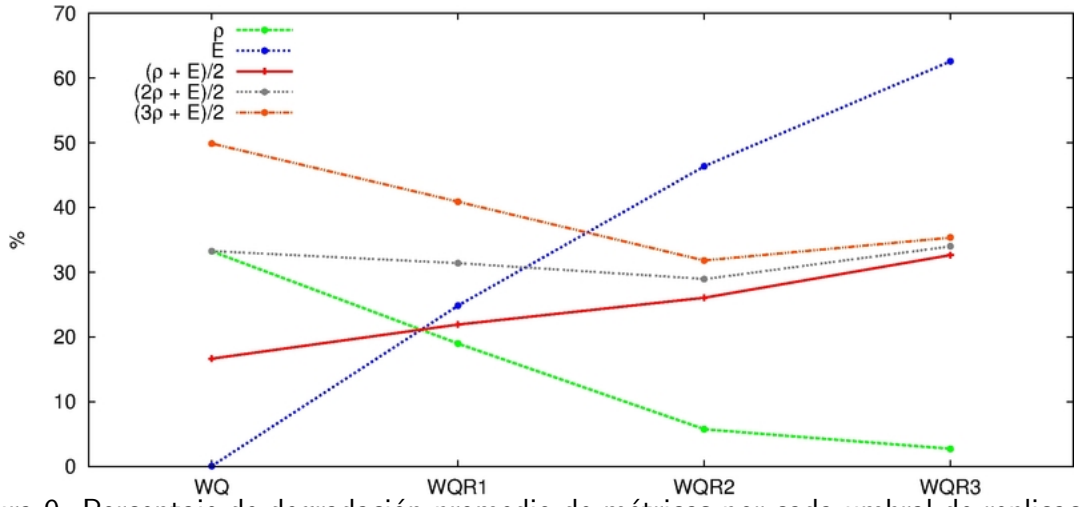

Figura 9. Porcentaje de degradación promedio de métricas por cada umbral de replicación.

Como se mencionó en el capítulo anterior, para poder analizar los datos generados por el proceso de evaluación, sin los posibles efectos negativos de permitir que

Tabla 8. Porcentaje de degradación promedio de métricas por cada umbral de replicación.

Réplicas	ρ	E^{grid}	$\frac{\rho+E^{\text{grid}}}{2}$	$\frac{2\cdot\rho+E^{\text{grid}}}{2}$	$\frac{3\cdot\rho+E^{\text{grid}}}{2}$
0	33.23	0.068	16.65	33.27	49.89
1	18.98	24.85	21.92	31.41	40.90
2	5.76	46.37	26.06	28.94	31.83
3	2.71	62.60	32.66	34.02	35.37

pequeñas porciones de las instancias del problema con una alta desviación dominen las conclusiones basadas en promedios, se presentan los perfiles de desempeño de los cuatro umbrales de replicación, los cuales dependen de una función constante, discreta y no-decreciente (ecuación 12), que presenta la razón de desempeño $r = \frac{\text{valor de la métrica}}{\text{mejor valor de la métrica}}$ para un factor τ (Dolan y Moré, 2002; Dolan *et al.*, 2006).

Aun dando el doble de importancia a ρ, el algoritmo bajo los cuatro umbrales de replicación tiene perfiles similares. Se puede notar que WQ y WQR3 dominan, ésto se debe a que WQ tiene el mejor desempeño en cuanto a consumo de energía (E^{grid}) y WQR3 tiene el mejor desempeño en cuanto a factor de aproximación (ρ). Ambos son seguidos de cerca por WQR2 que combina ambas métricas de mejor manera que WQR1, como se ve en la Figura 10.

Sin embargo, como se observa en la Figura 11 y en la Tabla 9, al enfocarnos sólo en el grid con más heterogeneidad, el Grid 5, se observa que la reducción en el tiempo de ejecución o la mejora en la QoS es lo suficientemente representativa para justificar el uso de réplicas. Esto se debe a que la mejora en QoS causada por el uso de réplicas es mayor que el aumento en el consumo de energía causado por las mismas, dando como resultado que WQR3 sea el mejor algoritmo, seguido de cerca por WQR2, después WQR1 y WQ al último.

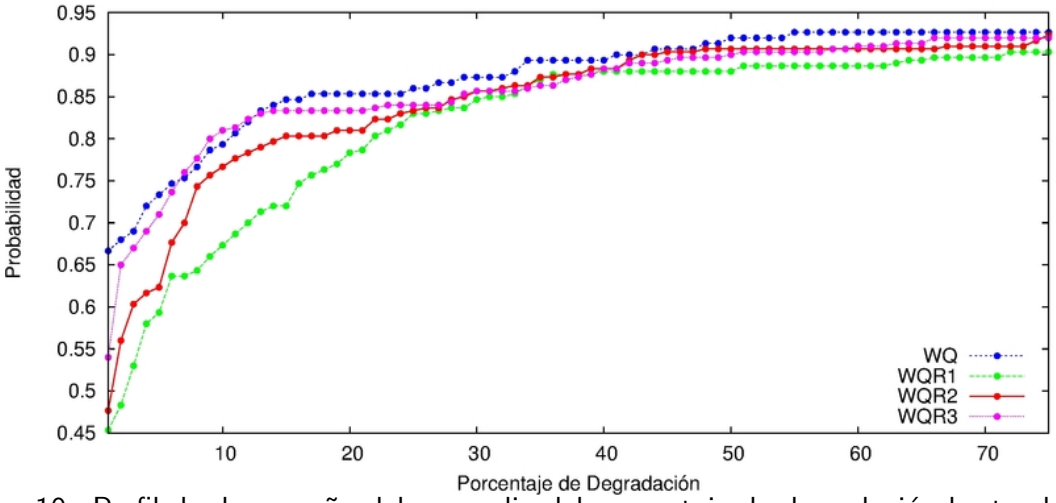

Figura 10. Perfil de desempeño del promedio del porcentaje de degradación hasta el 75% del promedio de energía y el doble del factor de aproximación.

Tabla 9. Degradación promedio de métricas por cada umbral de replicación en el Grid 5.

Réplicas	ρ	E^{grid}	$\frac{\rho+E^{\text{grid}}}{2}$
0	143.28	0.32	71.80
1	76.42	7.32	41.87
2	21.05	15.74	18.39
3	10.04	23.02	16.53

Esta diferencia también es notoria al calcular los perfiles de desempeño, como se muestra en la Figura 12, donde aun con la misma importancia, los algoritmos con réplicas tienen mejor desempeño que sin réplicas. Siendo WQR3 el mejor algoritmo, seguido de cerca por WQR2, después WQR1 y al final WQ.

Nótese que al la degradación depender de $\left(\frac{\text{valor de la métrica}}{\text{mejor valor de la métrica}} - 1\right) \cdot 100$, al ir aumentando (o empeorando) el mejor valor de la métrica, la degradación se reduce, pero en los valores absolutos los resultados son inferiores.

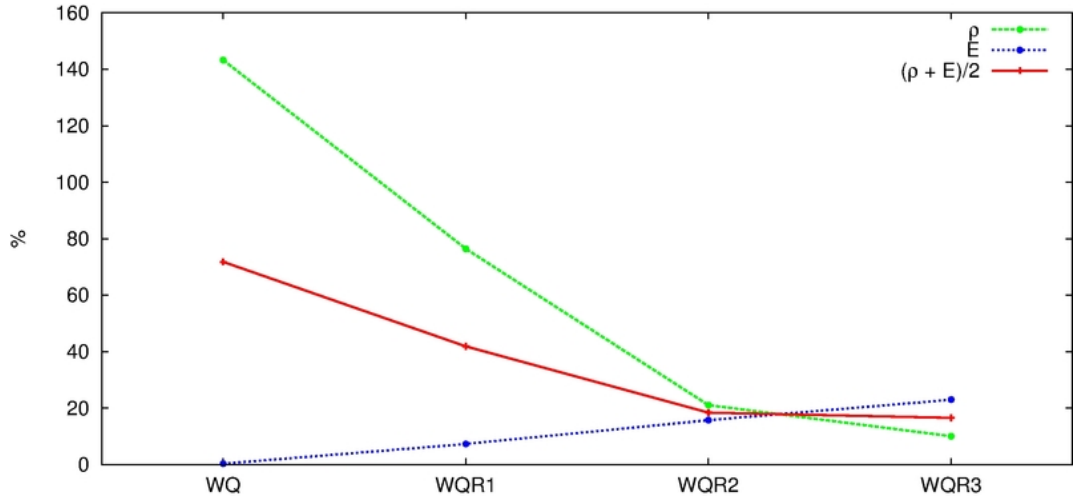

Figura 11. Degradación promedio de métricas por cada umbral de replicación en el Grid 5.

4.2.2 Costo de desempeño

Como se define en la ecuación 13 en el capítulo 3, el costo de desempeño es una métrica objetivo a minimizar que indica cuantas unidades de energía cuesta alcanzar una unidad de velocidad ($\frac{Wh}{D}$). Esta métrica al construirse con valores absolutos, no se ve afectada por los errores causados por la relatividad de la degradación.

Los resultados muestran que en promedio de todos los grids, no se justifica el uso de réplicas debido al exiguo aumento de velocidad en comparación con en el consumo de energía. Como se observa en la Figura 13, WQ da 84.39 debido a su mínimo consumo de energía; WQR1 aumenta a 95.7 debido a que aunque aumenta la velocidad, el aumento en el consumo de energía es más representativo; debido a que el aumento de velocidad en WQR2 es mayor mientras que el consumo de energía sigue aumentando de forma casi lineal, en este algoritmo se observa una mejora con respecto a WQR1 resultando en 94.67, pero en el caso de WQR3, se tiene el mayor consumo de energía, que aunque también se tiene la mayor velocidad, el aumento de ésta con respecto a WQR2 es mínimo, por lo que da 107.40.

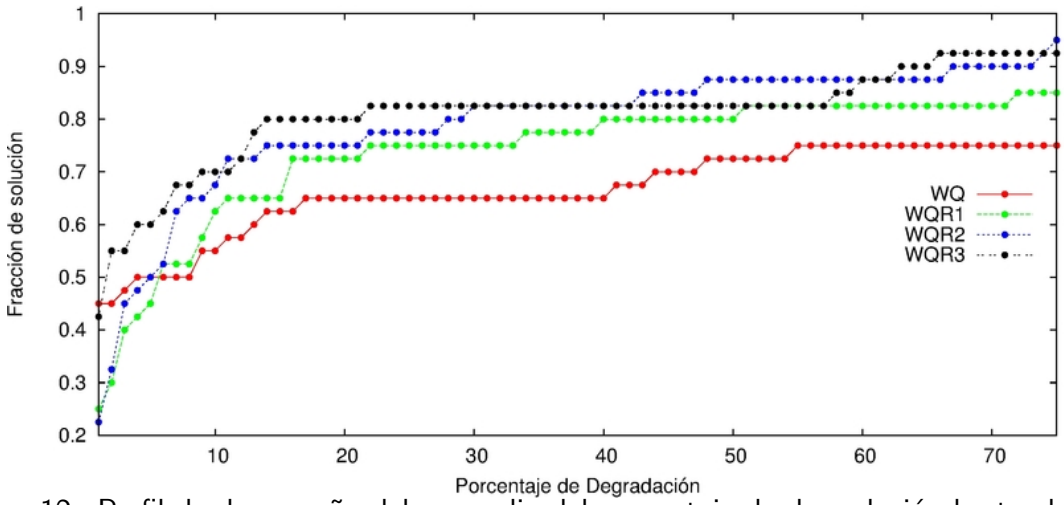

Figura 12. Perfil de desempeño del promedio del porcentaje de degradación hasta el 75% del promedio de energía y el factor de aproximación en el Grid 5.

Al igual que en el análisis de degradación, y como se observa en la Figura 14, al enfocarse sólo en el grid con mayor heterogeneidad, Grid 5, el costo de desempeño justifica el uso de réplicas. El algoritmo WQ es el que tiene mayor costo, resultando en 243.95, esto se debe al mal desempeño causado por malas asignaciones de tareas en un grid altamente heterogéneo. Al aplicar una réplica (WQR1) se obtiene una mejora significativa en el costo, dando 200.66. El algoritmo WQR3 logra alcanzar 129.19 dada su alta velocidad, pero el mejor resultado se obtuvo de WQR2, el cual cuesta 123.53.

Nótese que el mejor resultado obtenido en el Grid 5 es mas alto que el peor resultado obtenido en promedio, por lo que se puede inferir que al aumentar la heterogeneidad del grid empeora su calidad de servicio.

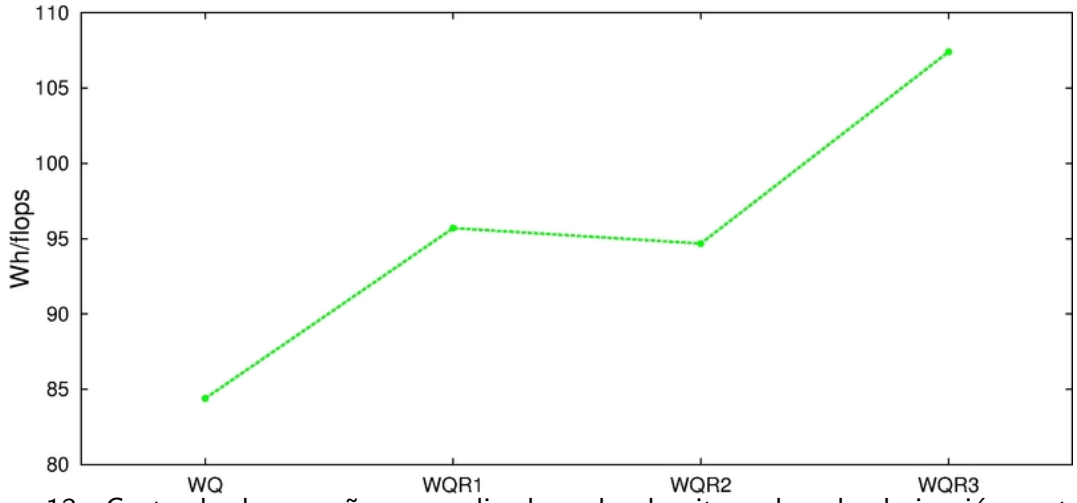

Figura 13. Costo de desempeño promedio de cada algoritmo de calendarización en todos los grids.

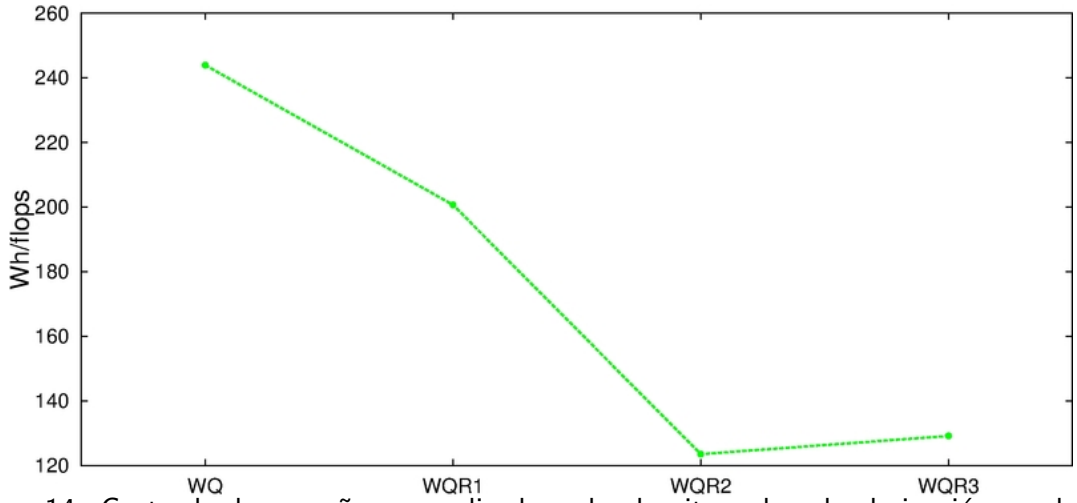

Figura 14. Costo de desempeño promedio de cada algoritmo de calendarización en el Grid 5.

Capítulo 5

Experimentos WQR-x

En este capítulo se extienden los experimentos presentados en la sección anterior al agregar heterogeneidad de eficiencia energética a los grids así como otro algoritmo libre de información y tres algoritmo de nivel de información 1. De igual manera, todos los experimentos fueron realizados utilizando tGSF descrito en Hirales-Carbajal *et al.* (2010). Se realizó un análisis de las distintas estrategias de calendarización en grids con heterogeneidad de velocidad y eficiencia energética de recursos con distintas BoTs que varían en su granularidad y número de operaciones de punto flotante promedio.

5.1 Diseño Experimental

A continuación se describen los dos aspectos mas relevantes para la evaluación de desempeño, cargas de trabajos representativas y un buen ambiente de pruebas, los cuales son necesarios para obtener resultados confiables, reproducibles y comparables. Nótese, que al fijar $S = \sum_{i=1}^{m} s_i$ y $p = \sum_{j=1}^{n} p_j$, las diferencias en los resultados dependen exclusivamente del algoritmo de calendarización.

5.1.1 Grids

Se consideran cinco escenarios de grid para la evaluación. Se fijó $S = \sum_{i=1}^{m} s_i = 1000$ para todos los grids. Los grids se diferencian por su heterogeneidad en velocidad y eficiencia energética, las cuales son dadas por la distribución uniforme $s_i = U\left(10 - \frac{hm}{2}, 10 + \frac{hm}{2}\right)$ (Silva *et al.*, 2003; Barrondo *et al.*, 2012) y la distribución uniforme $\text{eff}_i = U\left(10 - \frac{hm}{2}, 10 + \frac{hm}{2}\right)$, respectivamente. Para los cinco grids se usó $hm =$

$\{1, 2, 4, 8, 16\}$ obteniendo los valores que se muestran en las Figuras 6 y 15 y en el Apéndice A.2.

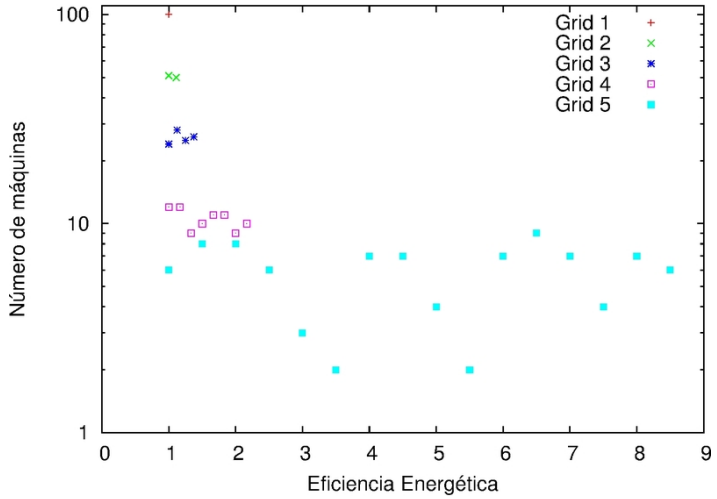

Figura 15. Histograma de eficiencia energética de los distintos grids.

La configuración de energía se adaptó de la propuesta en la bibliografía (Develder *et al.*, 2008; Ponciano y Brasileiro, 2010; Barrondo *et al.*, 2012), y se modificó para manejar valores relativos de eficiencia y velocidad para calcular el consumo ($e_i = \frac{s_i}{\text{eff}_i}$). Asignando el consumo de acuerdo con lo que se describe en la Tabla 10.

Tabla 10. Valores del modelo de energía usados para los experimentos de WQR-x.

Notación	Valor	Descripción
$P_i^{\text{núcleoTrabajo}}(t)$	10%	Potencia extra requerida cuando el núcleo trabaja
$P_i^{\text{núcleoPrendido}}(t)$	4%	Potencia requerida para mantener un núcleo en ocio
$P_i^{\text{máquinaPrendida}}(t)$	86%	Potencia requerida para mantener un núcleo en ocio
$P_i^{\text{prendeMáquina}}$	99%	Potencia requerido para prender una máquina
$T_i^{\text{prendeMáquina}}$	89 s	Tiempo requerido para prender una máquina
$T_i^{\text{apagaMáquina}}$	10 s	Tiempo de espera para apagar una máquina

5.1.2 Cargas de trabajo

Se consideran quince aplicaciones como cargas de trabajo, en las cuales todas se fijan en $p = \sum_{j=1}^{n} p_j = 3600000$. Las quince aplicaciones se dividen en tres grupos y cinco subgrupos que se caracterizan por diferente número de operaciones de punto flotante promedio ($\bar{p_j}$) de cada BoT. Los tres grupos se definen por $\bar{p_j} = \{5000, 25000, 125000\}$, lo que hace que varíe el número de tareas o granularidad del BoT, como se muestra en la Tabla 11 y la Figura 28. Los subgrupos consisten en la variación de $\bar{p_j}$ un 0%, 25%, 50%, 75% y 100%, con una distribución uniforme, al igual que en Silva *et al.* (2003).

De esta forma, se presentan diferentes escenarios de Grid con diferentes cargas de trabajo que varían de 36 a 0.29 tareas por máquina. Nótese que se eliminó uno de los grupos del capítulo anterior dado su bajo nivel de impacto.

Tabla 11. Relación entre tiempo de ejecución y número de tareas por máquina.

$\bar{p_j}$	n	Número de tareas por máquina
5000	720	7.2
25000	144	1.44
125000	29	0.29

5.2 Resultados

En promedio se confirma que la replicación sirve para minimizar el factor de aproximación (mejorar la QoS) de los algoritmos, siempre y cuando haya heterogeneidad en el grid. Se observa que los algoritmos con de nivel de información 1 tienen un mejor desempeño que los del nivel de información 0, como se observa en la Tabla 12 y en la Figura 16.

Tabla 12. Factor de aproximación promedio en todos los grids de los distintos algoritmos de calendarización.

Réplicas	WQR	WQR-h	WQR-s	WQR-eff	WQR-seff
0	2.05	2.05	1.56	1.56	1.56
1	1.65	1.63	1.33	1.36	1.35
2	1.46	1.46	1.30	1.33	1.31
3	1.4	1.39	1.29	1.33	1.31

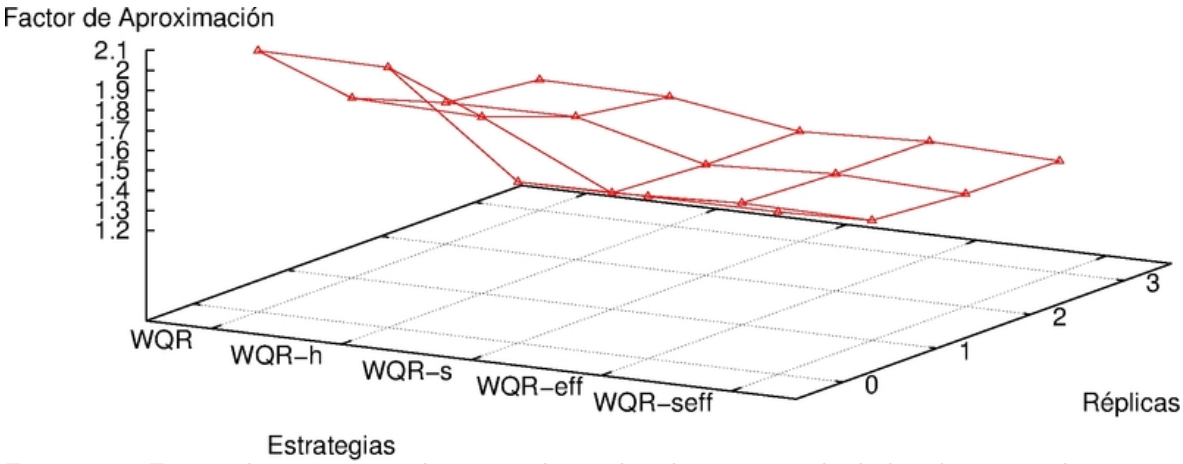

Figura 16. Factor de aproximación promedio en los distintos grids de los distintos algoritmos de calendarización.

Como se observa en la Figura 17 y en la Tabla 16, entre más heterogéneo es el grid, más es la mejora en la calidad de servicio dada por la replicación. Es importante notar que para grids homogéneos, como el Grid 1, es un ejercicio trivial, ya que todos los calendarios tienen el mismo desempeño sin importar la asignación, debido a que todas las máquinas son iguales, pero este experimento sirve para validar nuestro modelo. En los grids de baja heterogeneidad, como los Grids 2, 3 y 4, se observa una clara ventaja de los algoritmos del nivel de información 1 sobre los de nivel de información 0. Aunque en estos grids los la diferencia entre los algoritmos de nivel de información 0 y nivel de información 1 es mínima, ya que al todos los recursos son muy parecidos, por lo

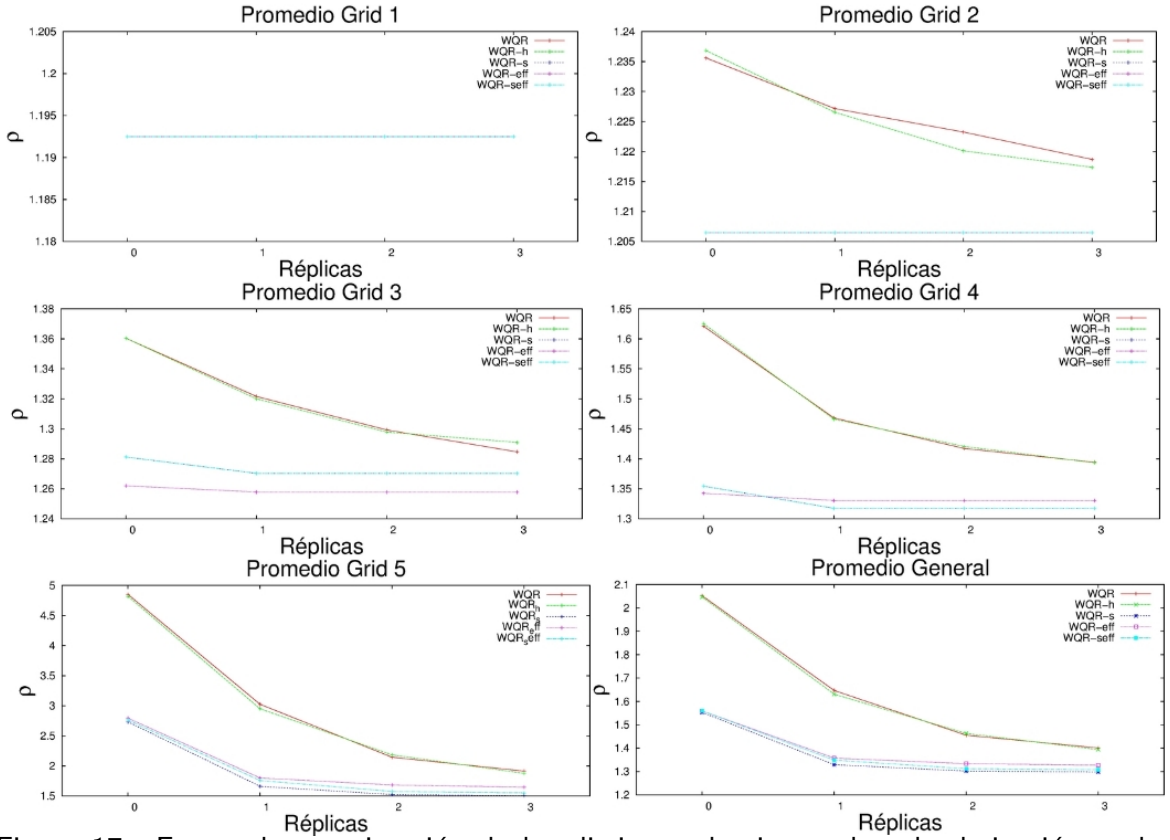

Figura 17. Factor de aproximación de los distintos algoritmos de calendarización en los distintos casos de estudio.

que el efecto de una mala asignación también es mínimo. Sin embargo, se encontró que la calidad de servicio de los algoritmos empeora al aumentar la heterogeneidad del sistema. Los P2P-DGs son sistemas altamente heterogéneos, aún más que el Grid 5. En este último encontramos la mayor mejora en QoS al utilizar replicación. Demostrando que los algoritmos de nivel de información 1 dan mejor QoS con menos réplicas que los de nivel de información 0.

Al igual que en el capítulo anterior, se observa que el uso de réplicas genera un aumento en el consumo de energía casi proporcional al número de réplicas, como se muestra en la Tabla 13 y en la Figura 18. Sin embargo, este consumo extra es menor en los algoritmos que usan información, esto se debe a que al seleccionar máquinas más energéticamente eficientes o simplemente más rápidas, ya que la reducción en el tiempo

de ejecucíon también influye en la reducción del consumo de energía.

Tabla 13. Consumo de energía (kWh) promedio en los distintos algoritmos de calendarización.

Réplicas	WQR	WQR-h	WQR-s	WQR-eff	WQR-seff
0	50.87	50.92	48.01	47.67	47.70
1	67.73	67.75	65.84	65.65	65.81
2	82.44	82.44	80.82	80.69	80.80
3	93.78	93.76	91.70	92.10	91.91

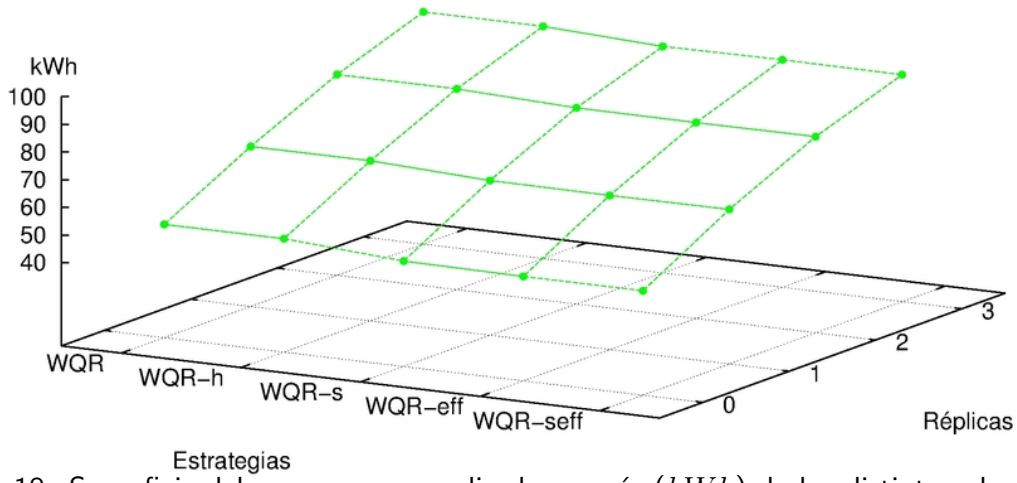

Figura 18. Superficie del consumo promedio de energía (kWh) de los distintos algoritmos de calendarización.

Al igual que con otras métricas, se puede observar que en un grid homogéneo no existe diferencia en el consumo eléctrico entre los algoritmos, pero de igual manera este caso valida nuestro modelo. En los grids con poca heterogeneidad, como lo son los Grid 2, 3 y 4, se nota poca diferencia en el consumo de energía entre los algoritmos que utilizan información y los algoritmos libres de información. Pero al aumentar la heterogeneidad, como en el caso del Grid 5, se observa una mayor diferencia en el consumo de energía entre los dos niveles de información, como se nota en la Figura

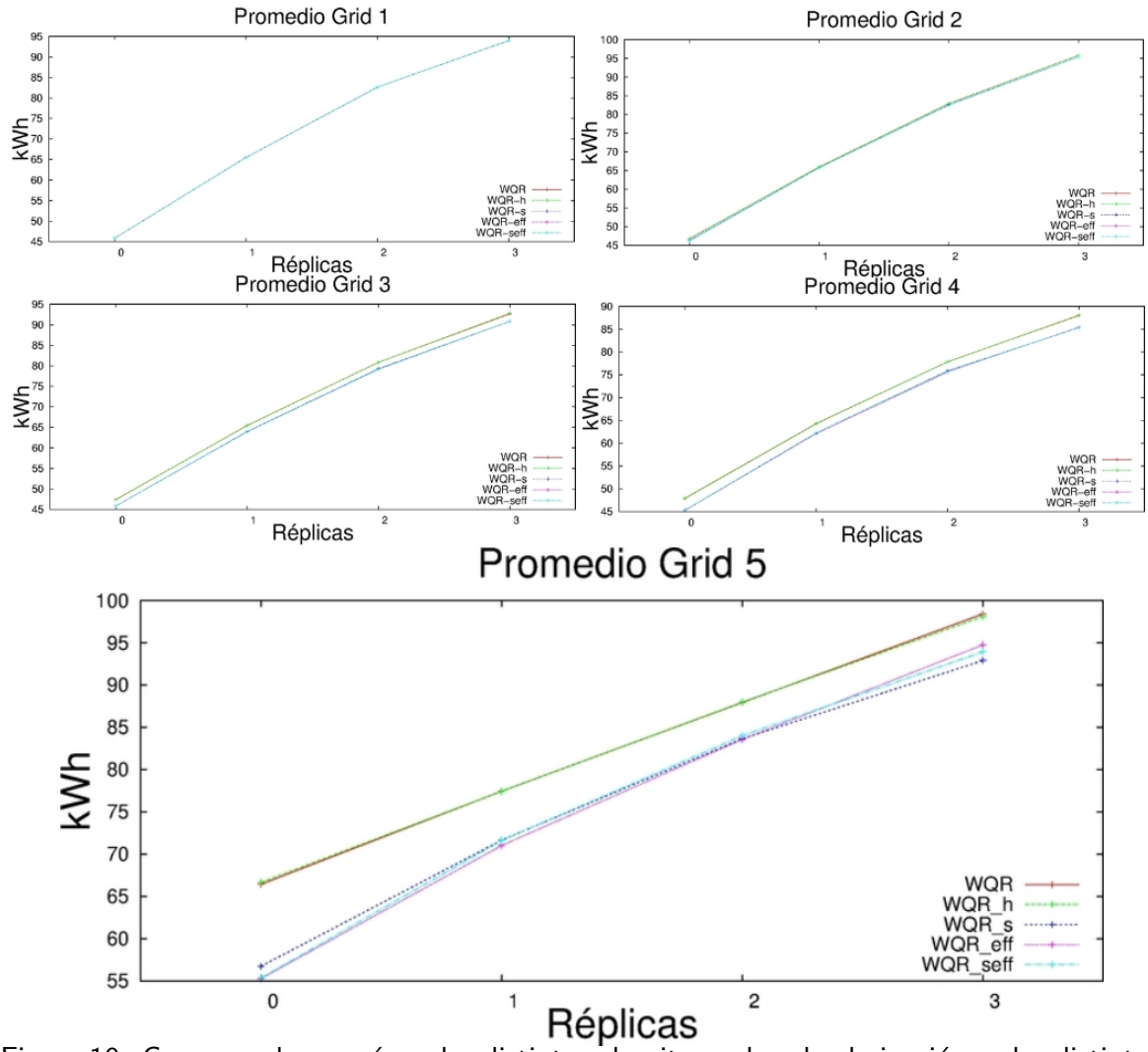

Figura 19. Consumo de energía en los distintos algoritmos de calendarización en los distintos grids.

19 y en la Tabla 17. Esta diferencia es causada por la toma de decisiones basadas en información, las cuales son mas acertadas que las decisiones aleatorias de los algoritmos libres de información.

5.2.1 Degradación

Debido a que los algoritmos de nivel de información 0 son superados en ambas métricas por los del nivel de información 1, es evidente que al promediar la degradacíon de

los valores de ambas métricas se observe una clara distinción entre los dos niveles de información, siendo los algoritmos del nivel de información 1 superiores en desempeño generalizado promedio que los del nivel de información 0, como podemos notar en la Tabla 14 y en la Figura 21.

Tabla 14. Degradación promedio en todos los grids del factor de aproximación y consumo de energía en los distintos algoritmos de calendarización.

Réplicas	WQR	WQR-h	WQR-s	WQR-eff	WQR-seff
0	32.49	32.37	7.93	7.86	7.79
1	37.18	36.68	21.87	22.83	22.55
2	46.46	46.62	38.04	39.01	38.39
3	57.25	56.91	50.32	51.85	50.86

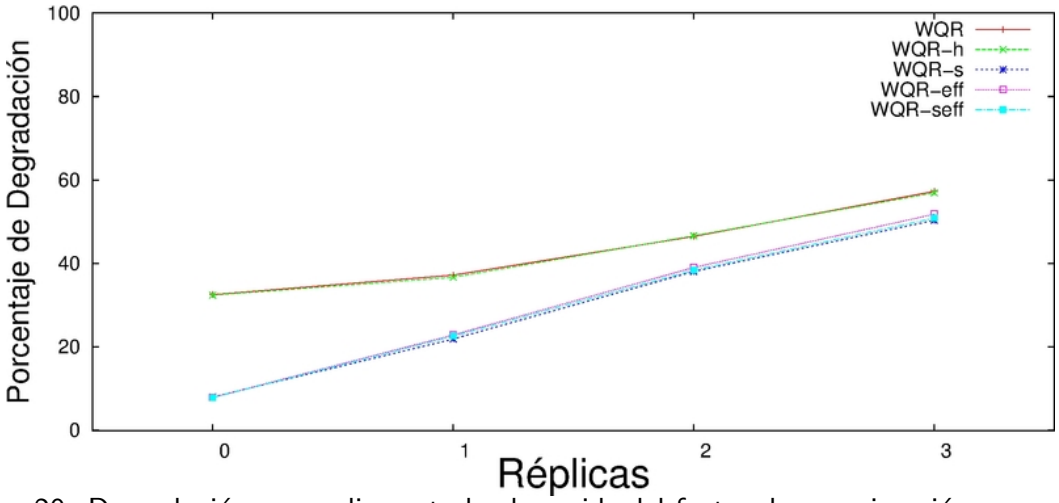

Figura 20. Degradación promedio en todos los grids del factor de aproximación y consumo de energía en los distintos algoritmos de calendarización.

Como muestra la Figura 21 y la Tabla 18, al evaluar el promedio de las degradaciones de los valores de las métricas en los cinco grids por separado se encuentra un comportamiento similar al del consumo de energía. Al ir aumentando la heterogeneidad en el grid, la diferencia entre los algoritmos del nivel de información 1 y los libres de información se hace más evidente. De hecho, en los algoritmos libres de información sin réplicas el aumento en la degradación parece ser proporcional al aumento en la heterogeneidad de máquina (con la excepción del Grid 5). Mas aparte, en los grids homogéneos y semi-homogéneos, estos resultados sugieren la inconveniencia del uso de réplicas, ya que debido a la similitud entre las máquinas, el aumento en la QoS no es lo suficientemente amplio para sobrellevar el aumento en el consumo de energía.

De cualquier manera, como se ha visto a lo largo de esta tesis, el Grid 5 es un caso excepcional debido a su alta heterogeneidad. De la misma forma que con otras métricas, en este caso resulta factible el uso de réplicas. El mejor desempeño generalizado en este caso lo obtuvo el algoritmo WQR-s 1 con un 23.92% de degradación (para mas detalles ver la Tabla 18).

Hay que hacer notar que al la degradación depender de $\left(\frac{\text{valor de la métrica}}{\text{mejor valor de la métrica}} - 1\right) \cdot 100$, una baja degradación no significa buen desempeño debido que si aumenta (o empeora) el mejor valor de la métrica, la degradación se reduce, pero en los valores absolutos se tiene un desempeño menor.

Para prevenir los efectos negativos de que pequeñas porciones de las instancias del problema con una alta desviación, como lo es un solo grupo de aplicaciones, dominen las conclusiones basadas en promedios, se presentan los perfiles de desempeño, de las veinte estrategias de calendarización evaluadas.

En general, como se muestra en la Figura 22, los algoritmos del nivel de información 1 proveen el mejor desempeño, teniendo menos de 1% de degradación en mas del 70%

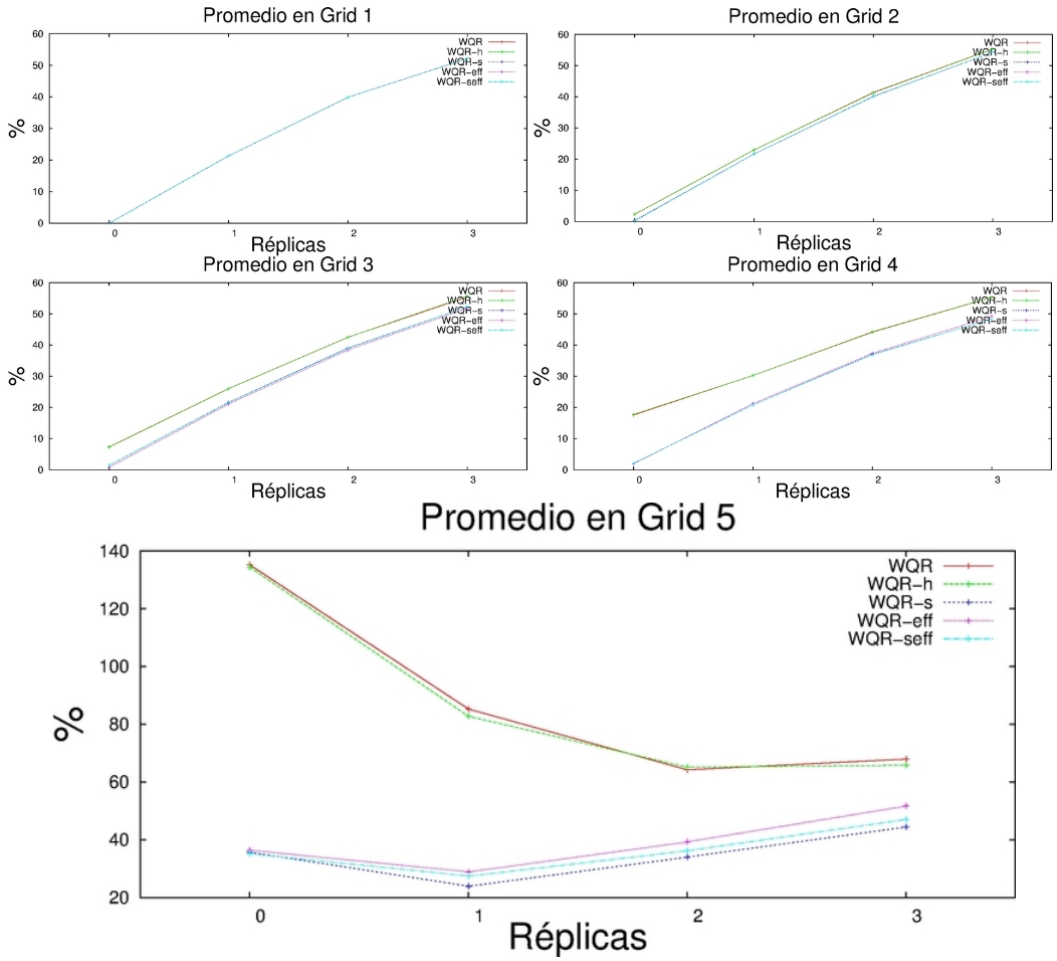

Figura 21. Degradación promedio en todos los grids del factor de aproximación y consumo de energía en los distintos algoritmos de calendarización.

de los resultados, haciendo innecesario el uso de réplicas.

Como se puede ver en la Figura 23, en los grids homogéneos y semi-homogéneos (Grids 1-4), dando igual importancia a ambas métricas para los perfiles de desempeño, los algoritmos de nivel de información 1 sin réplicas son claramente superiores al resto, con más del 70% de los resultados teniendo menor o igual que 1% de degradación (con la excepción de WQR-eff en el Grid 4 que es el 63% de las instancias). Esto se debe a la gran diferencia de degradación en consumo de energía entre los algoritmos con réplicas y los algoritmos sin réplicas, aún cuando los primeros tienen una mejor calidad

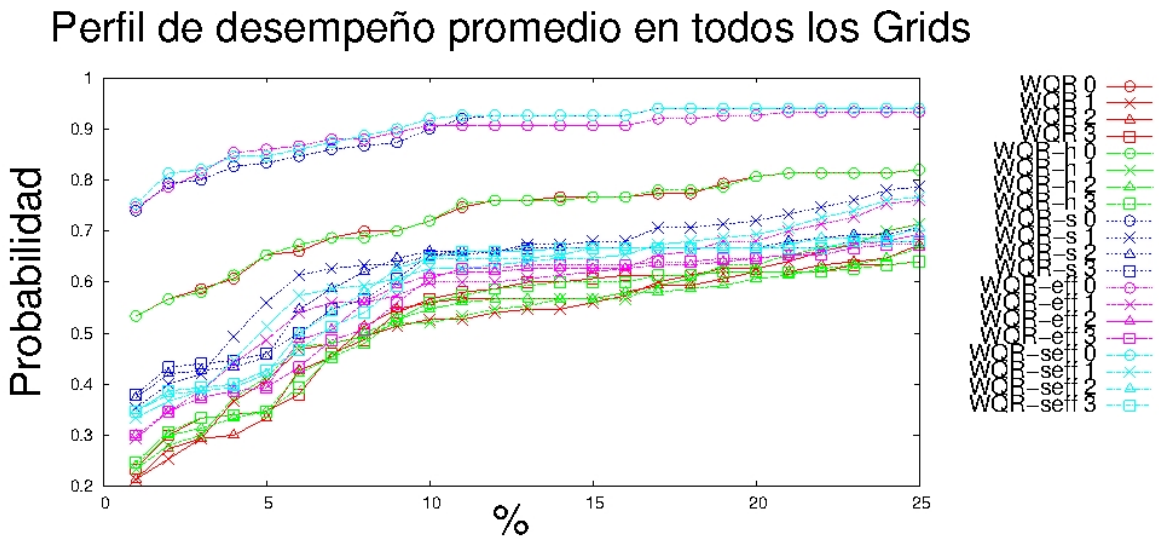

Figura 22. Perfiles de desempeño promedio de todos los grids.

de servicio, en ambientes poco heterogéneos la diferencia de calidad de servicio no es suficiente para justificar el uso de réplicas.

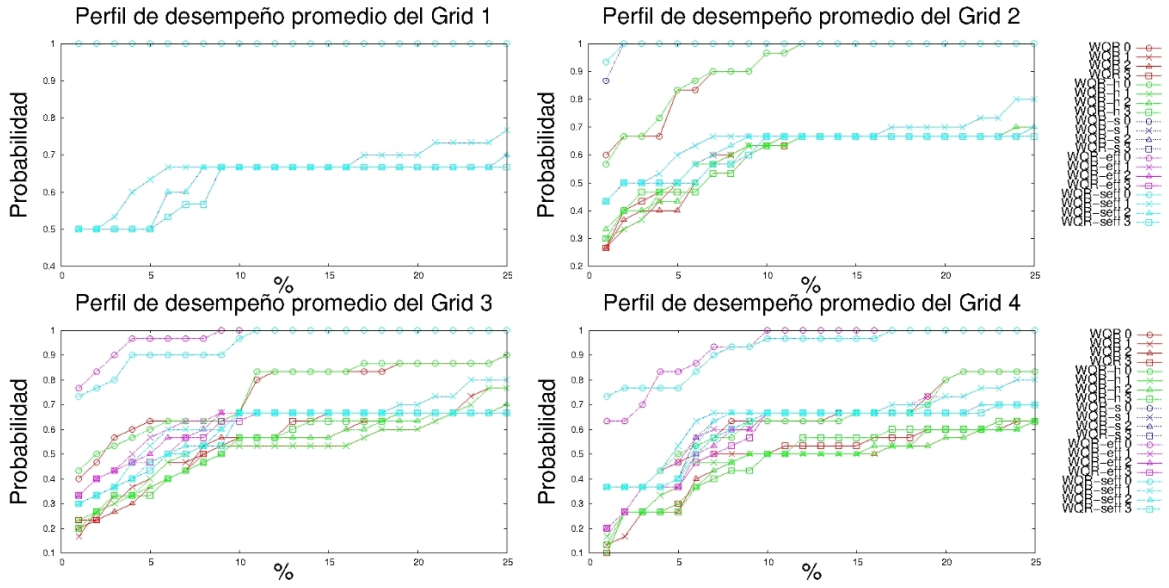

Figura 23. Perfiles de desempeño del por Grids.

Como se observa en la Figura 24, en un grid heterogéneo como el Grid 5, WQR-s 1 tiene una eficiencia similar a los algoritmos de nivel 1 sin réplicas, teniendo un 70% de las soluciones por debajo del 13% de degradación, siendo superior al resto de los algoritmos

con réplicas en este escenario. Pero aún en el Grid 5, el 50% de las soluciones dadas por los algoritmos de nivel de información 1 sin réplicas mantienen una degradación menor al 5%, lo cual evita definir cual de los cuatro algoritmos (WQR-s 0, WQR-eff 0, WQR-seff 0, WQR-s 1) es el mejor.

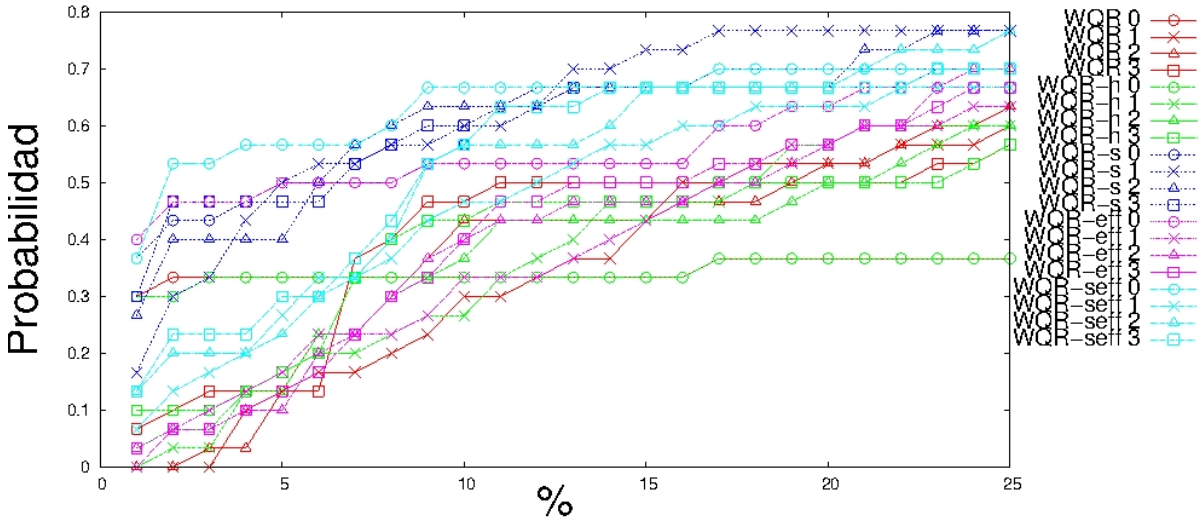

Figura 24. Perfiles de desempeño del Grid 5.

5.2.2 Costo de desempeño

Como se hace notar en la Tabla 15 y en la Figura 25, el costo de desempeño da resultados similares a los promedios de degradaciones, con la distinción que la diferencia entre los algoritmos libres de información y los de nivel de información 1 es mas marcada. Esto se debe al uso de valores absolutos en ves de relativos.

Al separar el costo de desempeño de los diferentes grids, se nota que en grids semi-homogéneos, como los el 2, 3 y 4 es menos necesario el uso de réplicas, ya que la mejora en el desempeño promedio del factor de aproximación en estos grids no “paga” el consumo extra de energía causado por el uso de replicación.

Tabla 15. Costo de desempeño ($\frac{Wh}{D}$) promedio de los distintos algoritmos de calendarización en todos los grids.

Réplicas	WQR	WQR-h	WQR-s	WQR-eff	WQR-seff
0	250.88	251.35	133.29	133.44	132.60
1	281.91	280.02	188.95	192.88	190.54
2	311.00	313.18	250.21	256.57	251.55
3	353.42	348.81	298.95	308.58	301.40

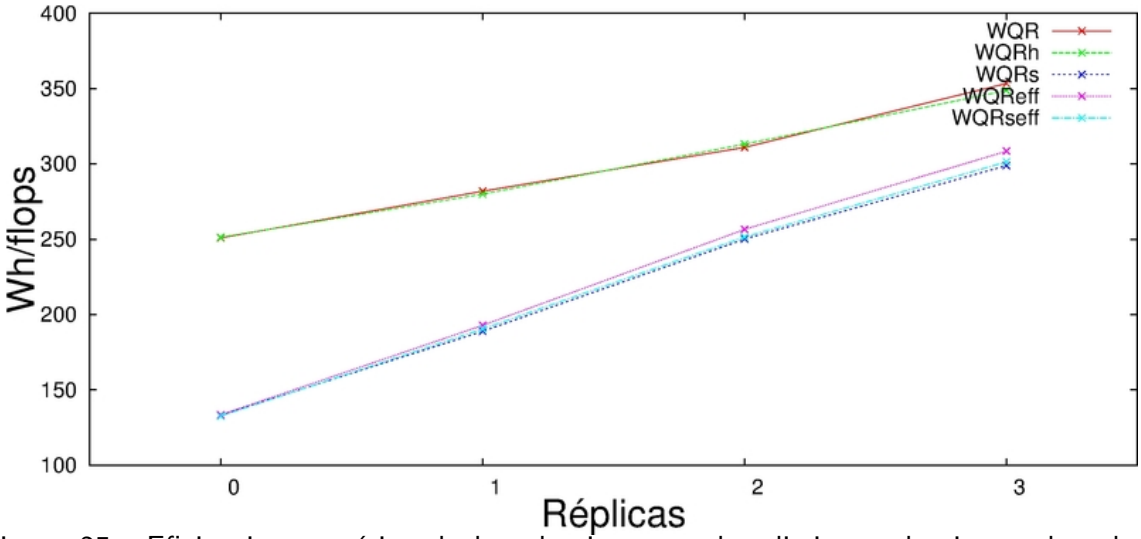

Figura 25. Eficiencia energética de los algoritmos en los distintos algoritmos de calendarización.

Sin embargo, en ambientes altamente heterogéneos, el costo de desempeño parece justificar este mecanismo de mejora de tiempo de ejecución, en el caso de los algoritmos del nivel de información 1, una réplica da el mejor desempeño. También se resalta la mala eficiencia de los algoritmos libres de información, aún así el uso réplicas mejora su desempeño en los sistemas altamente heterogéneos, tales como el Grid 5, como se muestra en la Figura 26 y en la Tabla 19.

Estos resultados confirman que el uso de información para la calendarización da un mejor desempeño que el uso de algoritmos libres de información, resultado contrario

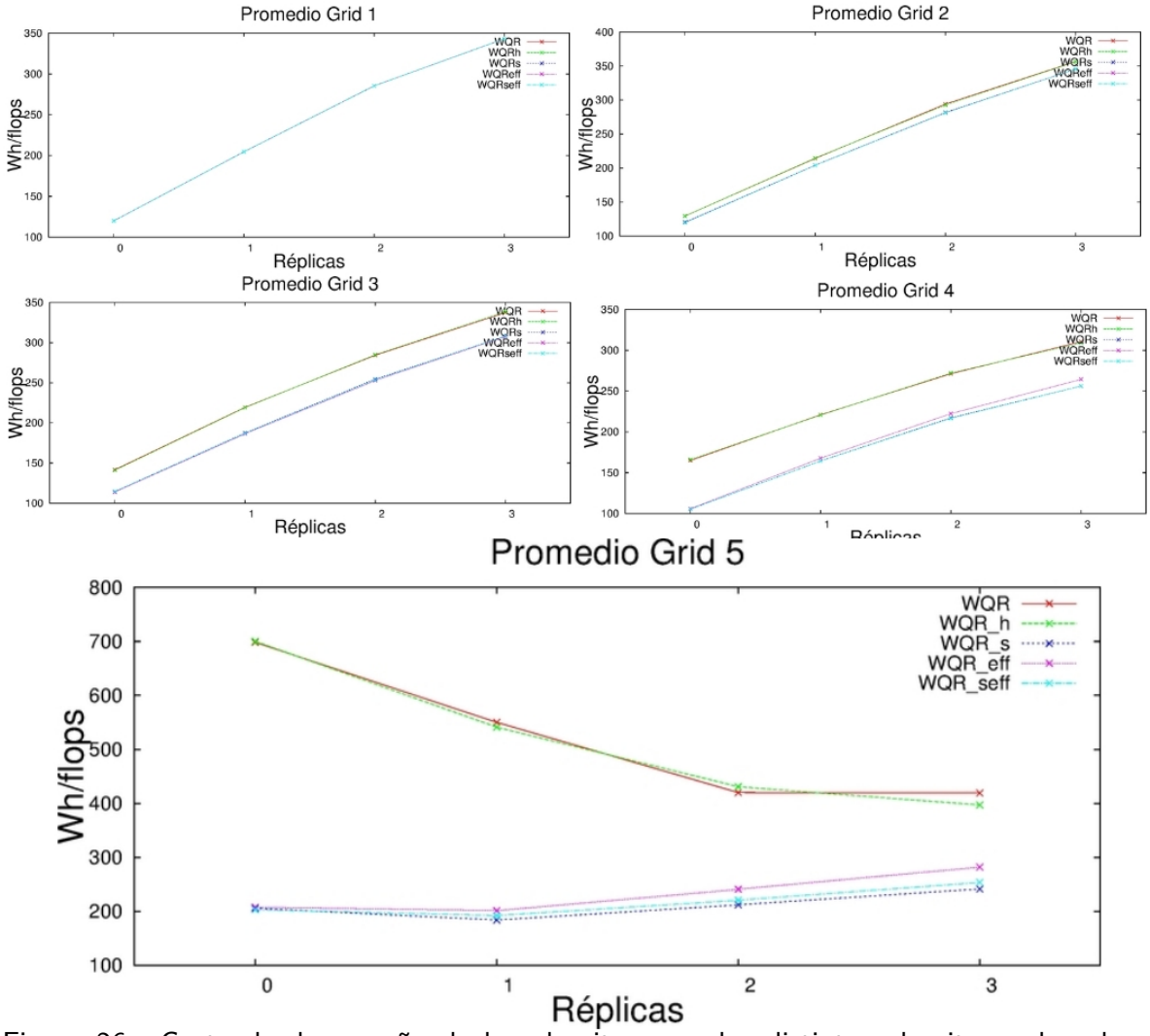

Figura 26. Costo de desempeño de los algoritmos en los distintos algoritmos de calendarización.

a lo encontrado en Silva *et al.* (2003). Además, también se comprueba que el uso de eficiencia energética como heurística de calendarización da buenos resultados en cuanto a eficiencia energética y desempeño, aunque estos datos son difíciles de conseguir en un grid real (Ponciano y Brasileiro, 2010). Se encontró que el sólo uso de la velocidad de las máquinas como heurística de calendarización da tan buenos resultados como el uso de eficiencia energética, esto se debe al casi perfecto balanceo de cargas sobre las máquinas con mayor velocidad. Finalmente, se demuestra que el uso de réplicas no es

necesario teniendo información de las máquinas del grid.

En esta tesis se trata uno de los problemas principales que enfrentan los Grids de Computadoras Personales (P2P-DGs, por sus siglas en inglés) para poder convertirse en una alternativa de bajo costo y viable para proveer de grandes capacidades de cómputo a centros de investigación y laboratorios pequeños. El problema de calendarización con buen desempeño y energéticamente eficiente. Por lo que el objetivo es examinar el desempeño energético y de calidad de servicio de distintos algoritmos de administración de recursos y posteriormente recomendar el que atienda los requerimientos meta, justificando la realización de este trabajo.

Posteriormente, en el capítulo 2 se establece el estado del arte de los P2P-DGs, especificando los distintos problemas abiertos:

(a) Topologías

(b) Transferencia de Datos

(c) Cómputo en Nube sobre esta infraestructura

(d) Calendarización

También se establece el estado del arte de las distintas estrategias para el manejo de energía en los distintos tipos de sistemas de cómputo distribuido:

(a) Clusters

(b) Grids

(c) Nubes de cómputo

haciendo énfasis especial en los P2P-DGs. Finalizando el estado del arte con resultados teóricos y experimentales de calendarización.

Después se describen los modelos de P2P-DGs, así como su consumo de energía, las respectivas cargas de trabajo (Bolsas de Tareas), las estrategias de calendarización a ser usadas y las métricas de evaluación. Dentro de dichas métricas, en esta tesis se introduce la métrica objetivo "costo de velocidad", la cual combina el tiempo de ejecución (o QoS) y el consumo de energía para indicar el número de unidades de energía (Wh) que se requieren para alcanzar una unidad de desempeño (D). Dichos modelos, estrategias y métricas dan pie a la experimentación con base en simulación de dos casos diferentes.

El primer caso o conjunto de experimentos repite los escenarios presentados en Silva *et al.* (2003), donde se evalúa el algoritmo WQR y la heterogeneidad es exclusiva a la velocidad de las máquinas, añadiendo el modelo de energía para justipreciar esa métrica y su relación con la calidad de servicio. Se encuentran deficiencias en el método aditivo de reducción de objetivos (degradaciones). También se hace un análisis de costo de velocidad. Los resultados de este primer conjunto de experimentos muestran que las réplicas no son necesarias a menos que el grid sea heterogéneo o que la reducción del tiempo de ejecución (o mejora de la QoS) tenga mayor importancia que la reducción el consumo de energía.

El segundo conjunto de experimentos extiende ese escenario introduciendo heterogeneidad en la eficiencia energética al ambiente de grid simulado. De la misma manera, se introducen otros algoritmos de calendarización con el objetivo de obtener otros puntos de comparación. Los algoritmos agregados al nivel de información 0 es, WQR-h que extiende WQR tomando decisiones con respecto a un historial basado en el Axioma 1. Así mismo, se incorpora un primer nivel de información, información de las máquinas,

este nivel incluye los algoritmos WQR-s, WQR-eff y WQR-seff, los cuales extienden WQR calendarizando las tareas a la máquina disponible más rápida, energéticamente eficiente y combinación de ambas, respectivamente.

Se observa que la solo existe una relación entre calidad de servicio y consumo de energía cuando el primero se obtiene a través del uso de réplicas. Se demuestra que al asignar igual importancia a ambas métricas, el uso de información de las máquinas del sistema para la calendarización (WQR-s, WQR-eff, WQR-seff) da mejor resultado que los algoritmos libres de información (WQR, WQR-h), teniendo más del 70% de las soluciones una degradación menor o igual al 1% conforme a la mejor solución encontrada. Aunque se pudiera discutir que en grids heterogéneos es útil el uso de réplicas, se encontró que no son necesarias siempre y cuando se tenga al menos un nivel de información para la calendarización, sea velocidad o eficiencia energética de las máquinas que conforman el grid, ya que ésta información mejora el desempeño de la calendarización sin incrementar el consumo de energía. Esto se debe a que el balanceo de cargas sobre las máquinas más rápidas o energéticamente eficientes, brinda el mejor compromiso entre consumo energético y calidad de servicio.

Este trabajo puede dar pie a trabajos que incluyan otros aspectos importantes de los P2P-DGs, como los es la latencia de red y la tolerancia a fallas. También debe buscarse el punto medio de heterogeneidad de máquinas donde las réplicas se empiezan a hacer necesarias. Finalmente, este trabajo trata de impulsar la construcción de P2P-DGs para experimentación con resultados más contundentes, así como encontrar y solucionar otros problemas que pudieran tener estas infraestructuras, con la finalidad de estimular su uso.

Tabla 16. Factor de aproximación en los distintos algoritmos de calendarización en los distintos Grids.

Réplicas	WQR	WQR-h	WQR-s	WQR-eff	WQR-seff
Promedio de Grid 1					
0	1.19	1.19	1.19	1.19	1.19
1	1.19	1.19	1.19	1.19	1.19
2	1.19	1.19	1.19	1.19	1.19
3	1.19	1.19	1.19	1.19	1.19
Promedio de Grid 2					
0	1.24	1.23	1.20	1.20	1.20
1	1.22	1.22	1.20	1.20	1.20
2	1.22	1.22	1.20	1.20	1.20
3	1.21	1.21	1.20	1.20	1.20
Promedio de Grid 3					
0	1.36	1.36	1.28	1.26	1.28
1	1.32	1.32	1.27	1.25	1.27
2	1.29	1.29	1.27	1.25	1.27
3	1.28	1.29	1.27	1.25	1.27
Promedio de Grid 4					
0	1.62	1.62	1.35	1.34	1.35
1	1.46	1.46	1.31	1.33	1.31
2	1.41	1.42	1.31	1.33	1.31
3	1.39	1.39	1.31	1.33	1.31
Promedio de Grid 5					
0	4.85	4.81	2.73	2.79	2.76
1	3.02	2.95	1.66	1.79	1.75
2	2.14	2.18	1.52	1.68	1.57
3	1.91	1.87	1.50	1.64	1.55

Tabla 17. Consumo de energía promedio (kWh) de los distintos algoritmos de calendarización en los distintos Grids.

Réplicas	WQR	WQR-h	WQR-s	WQR-eff	WQR-seff
Promedio de Grid 1					
0	45.86	45.86	45.86	45.86	45.86
1	65.51	65.51	65.51	65.51	65.51
2	82.63	82.63	82.63	82.63	82.63
3	93.98	93.98	93.98	93.98	93.98
Promedio de Grid 2					
0	46.79	46.78	46.31	46.16	46.16
1	65.97	65.99	65.76	65.74	65.74
2	82.96	82.94	82.60	82.50	82.50
3	95.82	95.82	95.42	95.43	95.43
Promedio de Grid 3					
0	47.39	47.34	45.84	45.83	45.83
1	65.43	65.47	63.99	63.91	63.90
2	80.84	80.83	79.31	79.12	79.14
3	92.67	92.80	90.85	90.85	90.86
Promedio de Grid 4					
0	47.87	47.94	45.29	45.24	45.31
1	64.27	64.32	62.26	62.07	62.26
2	77.84	77.82	75.87	75.61	75.74
3	88.00	88.09	85.34	85.45	85.35
Promedio de Grid 5					
0	66.43	66.66	56.74	55.26	55.36
1	77.45	77.45	71.67	71.04	71.62
2	87.93	87.98	83.66	83.58	84.01
3	98.42	98.12	92.92	94.78	93.94

Tabla 18. Porcentaje de degradación promedio del factor de aproximación y consumo de energía de los distintos algoritmos de calendarización en los distintos Grids.

Réplicas	WQR	WQR-h	WQR-s	WQR-eff	WQR-seff
Promedio de Grid 1					
0	5.e-15	3.e-15	6.e-15	6.e-15	6.e-15
1	21.32	21.32	21.32	21.32	21.32
2	39.87	39.87	39.87	39.87	39.87
3	52.18	52.18	52.18	52.18	52.18
Promedio de Grid 2					
0	2.31	2.35	0.35	0.18	0.18
1	22.95	22.97	21.67	21.64	21.64
2	41.40	41.26	40.17	40.06	40.06
3	55.35	55.30	54.29	54.30	54.30
Promedio de Grid 3					
0	7.33	7.26	1.47	0.80	1.46
1	26.05	26.01	21.62	21.09	21.51
2	42.51	42.48	39.08	38.43	38.88
3	55.41	55.80	52.29	51.86	52.30
Promedio de Grid 4					
0	17.57	17.80	2.09	1.92	2.10
1	30.26	30.26	20.82	21.23	20.81
2	44.15	44.29	37.05	37.39	36.88
3	55.33	55.37	48.39	49.21	48.41
Promedio de Grid 5					
0	135.22	134.43	35.71	36.38	35.20
1	85.28	82.80	23.92	28.86	27.42
2	64.23	65.16	34.02	39.30	36.21
3	67.94	65.87	44.43	51.68	47.07

Tabla 19. Costo de desempeño ($\frac{Wh}{D}$) de los distintos algoritmos de calendarización en los distintos Grids.

Réplicas	WQR	WQR-h	WQR-s	WQR-eff	WQR-seff
Promedio de Grid 1					
0	120.00	120.00	120.00	120.00	120.00
1	204.64	204.64	204.64	204.64	204.64
2	285.40	285.40	285.40	285.40	285.40
3	343.06	343.06	343.06	343.06	343.06
Promedio de Grid 2					
0	129.50	129.44	120.56	119.80	119.80
1	214.21	214.77	204.22	204.11	204.11
2	294.07	292.71	281.75	281.23	281.23
3	356.68	356.36	345.69	345.71	345.71
Promedio de Grid 3					
0	141.71	141.01	114.75	113.81	114.72
1	219.47	219.01	187.57	186.48	187.15
2	284.24	284.98	254.51	252.85	253.71
3	337.40	338.60	308.25	307.53	308.33
Promedio de Grid 4					
0	164.92	166.22	105.36	105.93	105.35
1	221.03	220.54	164.46	167.82	164.41
2	271.18	272.16	217.25	222.39	216.67
3	310.67	309.08	256.22	264.50	256.28
Promedio de Grid 5					
0	698.29	700.07	205.76	207.67	203.11
1	550.19	541.16	183.85	201.35	192.38
2	420.12	430.66	212.15	240.98	220.73
3	419.28	396.97	241.54	282.13	253.61

Referencias Bibliográficas

Allalouf, M., Arbitman, Y., Factor, M., Kat, R. I., Meth, K., y Naor, D. (2009). Storage modeling for power estimation. En *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference*, SYSTOR '09, páginas 3:1–3:10, New York, NY, USA. ACM. ISBN 978-1-60558-623-6.

Andersen, D. G., Franklin, J., Kaminsky, M., Phanishayee, A., Tan, L., y Vasudevan, V. (2009). FAWN: a fast array of wimpy nodes. En *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, SOSP '09, páginas 1–14, New York, NY, USA. ACM. ISBN 978-1-60558-752-3.

Anderson, D. P. (2004). BOINC: A system for public-resource computing and storage. En *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing.*, GRID '04, páginas 4–10, Washington, DC, USA. IEEE Computer Society. ISBN 0-7695-2256-4.

Anderson, D. P., Cobb, J., Korpela, E., Lebofsky, M., y Werthimer, D. (2002). Seti@home: an experiment in public-resource computing. *Commun. ACM*, **45**(11): 56–61.

Anglano, C. y Canonico, M. (2005). Fault-tolerant scheduling for bag-of-tasks grid applications. En *Proc. of the 2005 European Grid Conference (EuroGrid 2005). Lecture Notes in Computer Science*, página 630. Springer.

Anglano, C. y Canonico, M. (2008). Scheduling algorithms for multiple bag-of-task applications on desktop grids: A knowledge-free approach. En *IPDPS'08*, páginas 1–8.

Anglano, C., Canonico, M., y Guazzone, M. (2010). The sharegrid peer-to-peer desktop grid: Infrastructure, applications, and performance evaluation. *J. Grid Comput.*, **8**(4): 543–570.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., y Zaharia, M. (2009). Above the clouds: A berkeley view of cloud computing. Reporte técnico.

Awan, M. S. K. y Jarvis, S. A. (2012). Malikstone a contextaware lightweight benchmark for p2p computing. En *International Conference on High Performance Computing & Simulation (HPCS 2012)*. 3rd International Workshop on Modeling and Simulation of Peer-to-Peer Architectures and Systems (MOSPAS 2012).

Barrondo, A., Tchernykh, A., Schaeffer, E., y Pecero, J. (2012). Energy efficiency of knowledge-free scheduling in peer-to-peer desktop grids. En *International Conference*

on High Performance Computing & Simulation (HPCS 2012). 2012 Workshop on Optimization Issues in Energy Efficient Distributed Systems (OPTIM 2012).

Beloglazov, A., Buyya, R., Lee, Y. C., y Zomaya, A. Y. (2010). A taxonomy and survey of energy-efficient data centers and cloud computing systems. *CoRR*, **abs/1007.0066**.

Berral, J. L., Goiri, I. n., Nou, R., Julià, F., Guitart, J., Gavaldà, R., y Torres, J. (2010). Towards energy-aware scheduling in data centers using machine learning. En *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, e-Energy '10, páginas 215–224, New York, NY, USA. ACM. ISBN 978-1-4503-0042-1.

Buyya, R., Cortes, T., y Jin, H. (2001). Single system image. *Int. J. High Perform. Comput. Appl.*, **15**(2): 124–135.

Cappello, F., Desprez, F., Dayde, M., Jeannot, E., Jégou, Y., Lanteri, S., Melab, N., Namyst, R., Primet, P., Richard, O., Caron, E., Leduc, J., y Mornet, G. (2005). Grid'5000: a large scale, reconfigurable, controlable and monitorable grid platform. En *6th IEEE/ACM International Workshop on Grid Computing - GRID 2005*, Seattle, USA, États-Unis. Grid 2005 held in conjunction with SC'05, the International Conference for High Performance Computing, Networking and Storage.

Caulfield, A. M., Grupp, L. M., y Swanson, S. (2009). Gordon: using flash memory to build fast, power-efficient clusters for data-intensive applications. *SIGPLAN Not.*, **44**(3): 217–228.

Chakravarti, A. J., Baumgartner, G., y Lauria, M. (2004). The organic grid: Self-organizing computation on a peer-to-peer network. En *IEEE Transactions on Systems, Man, and Cybernetics*, páginas 96–103.

Chase, J. S., Anderson, D. C., Thakar, P. N., Vahdat, A. M., y Doyle, R. P. (2001). Managing energy and server resources in hosting centers. *SIGOPS Oper. Syst. Rev.*, **35**(5): 103–116.

Chee, B. y Franklin, C. (2010). *Cloud Computing: Technologies and Strategies of the Ubiquitous Data Center*. Taylor and Francis. ISBN 9781439806128.

Chien, A., Calder, B., Elbert, S., y Bhatia, K. (2003). Entropia: architecture and performance of an enterprise desktop grid system. *J. Parallel Distrib. Comput.*, **63**(5): 597–610.

Choi, S., Kim, H., Byun, E., Baik, M., Kim, S., Park, C., y Hwang, C. (2007). Characterizing and classifying desktop grid. En *CCGRID '07: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*, páginas 743–748, Washington, DC, USA. IEEE Computer Society. ISBN 0-7695-2833-3.

Cirne, W., Brasileiro, F., Andrade, N., Costa, L., Andrade, A., Novaes, R., y Mowbray, M. (2006). Labs of the world, unite!!! *Journal of Grid Computing*, **4**(3): 225–246.

Costa, F., Silva, L. M., Fedak, G., y Kelley, I. (2008). Optimizing data distribution in desktop grid platforms. *Parallel Processing Letters*, **18**(3): 391–410.

Costa, G. D., Gelas, J.-P., Georgiou, Y., Lefèvre, L., Orgerie, A.-C., Pierson, J.-M., Richard, O., y Sharma, K. (2009). The GREEN-NET framework: Energy efficiency in large scale distributed systems. En *IPDPS*, páginas 1–8.

Cunsolo, V. D., Distefano, S., Puliafito, A., y Scarpa, M. (2009). Volunteer computing and desktop cloud: The cloud@home paradigm. En *Proceedings of the 2009 Eighth IEEE International Symposium on Network Computing and Applications*, NCA '09, páginas 134–139, Washington, DC, USA. IEEE Computer Society. ISBN 978-0-7695-3698-9.

Delaluz, V., Sivasubramaniam, A., Kandemir, M., Vijaykrishnan, N., y Irwin, M. J. (2002). Scheduler-based DRAM energy management. En *Proceedings of the 39th annual Design Automation Conference*, DAC '02, páginas 697–702, New York, NY, USA. ACM. ISBN 1-58113-461-4.

Develder, C., Pickavet, M., Dhoedt, B., y Demeester, P. (2008). A power-saving strategy for grids. En *Proc. 2nd Int. Conf. on Networks for Grid Applications (GridNets 2008)*.

Dolan, E. D. y Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, **91**: 201–213.

Dolan, E. D., Moré, J. J., y Munson, T. S. (2006). Optimality measures for performance profiles. *SIAM J. on Optimization*, **16**(3): 891–909.

Fan, X., Weber, W.-D., y Barroso, L. A. (2007). Power provisioning for a warehouse-sized computer. *SIGARCH Comput. Archit. News*, **35**(2): 13–23.

Feng, W.-c. y Cameron, K. (2007). The green500 list: Encouraging sustainable supercomputing. *Computer*, **40**(12): 50–55.

Foster, I. y Kesselman, C. (1999). *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. ISBN 1558604758.

Fujimoto, N. y Hagihara, K. (2003). Near-optimal dynamic task scheduling of independent coarse-grained tasks onto a computational grid. En *Proc. ICPP 2003*, páginas 391–398.

Galambos, G. y Woeginger, G. J. (1993). An on-line scheduling heuristic with better worst case ratio than graham's list scheduling. *SIAM J. Comput.*, **22**(2): 349–355.

Garey, M. R. y Johnson, D. S. (1978). " strong " NP-completeness results: Motivation, examples, and implications. *J. ACM*, **25**(3): 499–508.

Ge, R., Feng, X., y Cameron, K. W. (2005a). Improvement of power-performance efficiency for high-end computing. En *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 11 - Volume 12*, IPDPS '05, páginas 233.2–, Washington, DC, USA. IEEE Computer Society. ISBN 0-7695-2312-9.

Ge, R., Feng, X., y Cameron, K. W. (2005b). Performance-constrained distributed dvs scheduling for scientific applications on power-aware clusters. En *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, SC '05, páginas 34–, Washington, DC, USA. IEEE Computer Society. ISBN 1-59593-061-2.

Goodman, J. R. y Sequin, C. H. (1981). Hypertree: A multiprocessor interconnection topology. *IEEE Trans. Comput.*, **30**(12): 923–933.

Graham, R. L. (1966). Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, **45**(9): 1563–1581.

Graham, R. L. (1969). Bounds on multiprocessing timing anomalies. *SIAM JOURNAL ON APPLIED MATHEMATICS*, **17**(2): 416–429.

Graham, R. L., Lawler, E. L., Lenstra, J. K., y Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, **5**(2): 287–326.

Gunaratne, C., Christensen, K., y Nordman, B. (2005). Managing energy consumption costs in desktop pcs and lan switches with proxying, split tcp connections, and scaling of link speed. *Int. J. Netw. Manag.*, **15**(5): 297–310.

Hirales-Carbajal, A., Tchernykh, A., Röblitz, T., y Yahyapour, R. (2010). A grid simulation framework to study advance scheduling strategies for complex workflow applications. En *IPDPS Workshops*, páginas 1–8.

Hirales-Carbajal, A., Tchernykh, A., Yahyapour, R., González-García, J. L., Röblitz, T., y Ramírez-Alcaraz, J. M. (2012). Multiple workflow scheduling strategies with user run time estimates on a grid. *J. Grid Comput.*, **10**(2): 325–346.

Hlavacs, H., Costa, G. D., y Pierson, J.-M. (2009). Energy consumption of residential and professional switches. En *CSE (1)*, páginas 240–246. IEEE Computer Society.

Hochbaum, D. S. y Shmoys, D. B. (1987). Using dual approximation algorithms for scheduling problems theoretical and practical results. *J. ACM*, **34**(1): 144–162.

Hochbaum, D. s. y Shmoys, D. B. (1988). A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM J. Comput.*, **17**(3): 539–551.

Horowitz, E. y Sahni, S. (1976). Exact and approximate algorithms for scheduling nonidentical processors. *J. ACM*, **23**(2): 317–327.

Iosup, A., Dumitrescu, C., Epema, D., Li, H., y Wolters, L. (2006). How are real grids used? the analysis of four grid traces and its implications. En *Proceedings of the 7th IEEE/ACM International Conference on Grid Computing*, GRID '06, páginas 262–269, Washington, DC, USA. IEEE Computer Society. ISBN 1-4244-0343-X.

Jimeno, M. y Christensen, K. (2008). A network connection proxy to enable hosts to sleep and save energy. IEEE.

Johnson, B. (2008). Cloud computing is a trap, warns GNU founder richard stallman. *The Guardian*.

Kim, K. H., Buyya, R., y Kim, J. (2007). Power aware scheduling of bag-of-tasks applications with deadline constraints on DVS-enabled clusters. En *Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*, CCGRID '07, páginas 541–548, Washington, DC, USA. IEEE Computer Society. ISBN 0-7695-2833-3.

Krashinsky, R. y Balakrishnan, H. (2002). Minimizing energy for wireless web access using bounded slowdown. En *8th ACM MOBICOM 2002*, Atlanta, GA.

Kurp, P. (2008). Green computing. *Commun. ACM*, **51**(10): 11–13.

Lenstra, J. K., Rinnooy Kan, A. H. G., y Brucker, P. (1977). *Complexity of Machine Scheduling Problems*, Vol. 1 de *Annals of Discrete Mathematics*, páginas 343–362. Elsevier. ISBN 9780720407655.

Lowekamp, B., Miller, N., Gross, T., Steenkiste, P., Subhlok, J., y Sutherland, D. (1999). A resource query interface for network-aware applications. *Cluster Computing*, **2**(2): 139–151.

Marcon, M., Viswanath, B., Cha, M., y Gummadi, K. P. (2011). Sharing social content from home: a measurement-driven feasibility study. En *Proceedings of the 21st international workshop on Network and operating systems support for digital audio and video*, NOSSDAV '11, páginas 45–50, New York, NY, USA. ACM. ISBN 978-1-4503-0777-2.

Merkel, A. y Bellosa, F. (2006). Balancing power consumption in multiprocessor systems. En *First ACM SIGOPS EuroSys Conference*, Leuven, Belgium.

Minas, L. y Ellison, B. (2009). *Energy Efficiency for Information Technology: How to Reduce Power Consumption in Servers and Data Centers*. Intel Press. ISBN 9781934053201.

Miriam, D. D. H. y Easwarakumar, K. S. (2011). HPGRID: a novel architectural model for resource management systems. En *Proceedings of the 2011 International Conference on Communication, Computing & Security*, ICCCS '11, páginas 427–432, New York, NY, USA. ACM. ISBN 978-1-4503-0464-1.

Morrison, J. P., John, S., Power, D. A., Cafferkey, N., y Patil, A. (2005). A grid application development platform for webcom-g. En *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*, Vol. 2, páginas 694–701 Vol. 2.

Murphy, M. A., Abraham, L., Fenn, M., y Goasguen, S. (2010). Autonomic clouds on the grid. *J. Grid Comput.*, **8**(1): 1–18.

Neary, M. O., Phipps, A., Richman, S., y Cappello, P. (2000). Javelin 2.0: Java-based parallel computing on the internet. En *Internet, Proceedings of European Parallel Computing Conference (Euro-Par 2000*, páginas 1231–1238. Springer.

Orgerie, A.-C., Lefèvre, L., y Gelas, J.-P. (2008). Save watts in your grid: Green strategies for energy-aware framework in large scale distributed systems. En *Proceedings of the 2008 14th IEEE International Conference on Parallel and Distributed Systems*, ICPADS '08, páginas 171–178, Washington, DC, USA. IEEE Computer Society. ISBN 978-0-7695-3434-3.

Orgerie, A.-C., Lefevre, L., y Gelas, J.-P. (2010). Demystifying energy consumption in grids and clouds. En *Proceedings of the International Conference on Green Computing*, GREENCOMP '10, páginas 335–342, Washington, DC, USA. IEEE Computer Society. ISBN 978-1-4244-7612-1.

Pierre, G. (2007). How cn and p2p technologies may help build next-generation grids. En *Proceedings of the second workshop on Use of P2P, GRID and agents for the development of content networks*, UPGRADE '07, páginas 25–28, New York, NY, USA. ACM. ISBN 978-1-59593-718-6.

Pinheiro, E., Bianchini, R., Carrera, E. V., y Heath, T. (2001). Load balancing and unbalancing for power and performance in cluster-based systems. En *Proceedings of the Workshop on Compilers and Operating Systems for Low Power (COLP'01)*.

Ponciano, L. y Brasileiro, F. V. (2010). On the impact of energy-saving strategies in opportunistic grids. En *GRID*, páginas 282–289. IEEE. ISBN 978-1-4244-9348-7.

Quezada-Pina, A., Tchernykh, A., González-García, J. L., Hirales-Carbajal, A., Miranda-López, V., Ramírez-Alcaraz, J. M., Schwiegelshohn, U., y Yahyapour, R. (2012). Adaptive job scheduling on hierarchical grids. *Future Generation Computer Systems*, **28/7**(1): 965–976.

Ramírez-Alcaraz, J. M., Tchernykh, A., Yahyapour, R., Schwiegelshohn, U., Quezada-Pina, A., González-García, J. L., y Hirales-Carbajal, A. (2011). Job allocation strategies with user run time estimates for online scheduling in hierarchical grids. *J. Grid Comput.*, **9**(1): 95–116.

Schwiegelshohn, U. y Tchernykh, A. (2012). Online scheduling for cloud computing and different service levels. En *IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, páginas 1061–1068.

Schwiegelshohn, U., Tchernykh, A., y Yahyapour, R. (2008). Online scheduling in grids. En *IPDPS*, páginas 1–10.

Sharma, K. y Aggarwal, S. (2009). Energy aware scheduling on desktop grid environment with static performance prediction. En *Proceedings of the 2009 Spring Simulation Multiconference*, SpringSim '09, páginas 105:1–105:8, San Diego, CA, USA. Society for Computer Simulation International.

Shmoys, D. B., Wein, J., y Williamson, D. P. (1995). Scheduling parallel machines on-line. *SIAM J. Comput.*, **24**(6): 1313–1331.

Silva, D. P. D., Cirne, W., Brasileiro, F. V., y Grande, C. (2003). Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids. En *Applications on Computational Grids, in Proc of Euro-Par 2003*, páginas 169–180.

Snowdon, D. C., Ruocco, S., y Heiser, G. (2005). Power management and dynamic voltage scaling: Myths and facts. En *Proceedings of the 2005 Workshop on Power Aware Real-time Computing*, New Jersey, USA.

Srikantaiah, S., Kansal, A., y Zhao, F. (2008). Energy aware consolidation for cloud computing. En *Proceedings of the 2008 conference on Power aware computing and systems*, HotPower'08, páginas 10–10, Berkeley, CA, USA. USENIX Association.

Tchernykh, A., Schwiegelshohn, U., Yahyapour, R., y Kuzjurin, N. (2010). On-line hierarchical job scheduling on grids with admissible allocation. *J. of Scheduling*, **13**(5): 545–552.

Tolentino, M. E., Turner, J., y Cameron, K. W. (2007). Memory-miser: a performance-constrained runtime system for power-scalable clusters. En *Proceedings of the 4th international conference on Computing frontiers*, CF '07, páginas 237–246, New York, NY, USA. ACM. ISBN 978-1-59593-683-7.

Torres, J., Carrera, D., Hogan, K., Gavalda, R., Beltran, V., y Poggi, N. (2008). Reducing wasted resources to help achieve green data centers. *Parallel and Distributed Processing Symposium, International*, **0**: 1–8.

Tsafrir, D., Etsion, Y., y Feitelson, D. G. (2007). Backfilling using system-generated predictions rather than user runtime estimates. *IEEE Trans. Parallel Distrib. Syst.*, **18**(6): 789–803.

Valentini, G. L., Lassonde, W., Khan, S. U., Min-Allah, N., Madani, S. A., Li, J., Zhang, L., Wang, L., Ghani, N., Kolodziej, J., y et al. (2011). An overview of energy efficiency techniques in cluster computing systems. *Cluster Computing*.

Vasić, N., Barisits, M., Salzgeber, V., y Kostic, D. (2009). Making cluster applications energy-aware. En *Proceedings of the 1st workshop on Automated control for datacenters and clouds*, ACDC '09, páginas 37–42, New York, NY, USA. ACM. ISBN 978-1-60558-585-7.

Weiss, A. (2007). Computing in the clouds. *netWorker*, **11**(4): 16–25.

Apéndice A

Configuración para los experimentos WQR

En este apéndice se presentan las distintas configuraciones de grid y cargas de trabajo para los experimentos presentados en el capítulo 4.

Grid 1 $G_s = \{$1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1$\}$

Grid 2 $G_s = \{$1, 1, 1.11, 1, 1, 1, 1.11, 1.11, 1.11, 1, 1, 1.11, 1.11, 1, 1.11, 1.11, 1.11, 1.11, 1.11, 1.11, 1.11, 1.11, 1, 1, 1.11, 1.11, 1.11, 1.11, 1.11, 1.11, 1, 1, 1, 1.11, 1, 1.11, 1.11, 1.11, 1, 1, 1, 1, 1.11, 1.11, 1.11, 1, 1.11, 1, 1.11, 1.11, 1.11, 1, 1.11, 1.11, 1, 1, 1.11, 1.11, 1.11, 1, 1, 1, 1, 1.11, 1.11, 1.11, 1, 1, 1.11, 1, 1, 1, 1.11, 1, 1, 1.11, 1, 1, 1.11, 1.11, 1, 1.11, 1, 1, 1.11, 1, 1.11, 1, 1, 1.11, 1.11, 1.11, 1, 1.11, 1, 1, 1, 1.11, 1, 1, 1, 1.11, 1, 1.11, 1, 1.11$\}$

Grid 3 $G_s = \{$1, 1.25, 1, 1, 1.25, 1.25, 1.125, 1.125, 1, 1.375, 1.375, 1.125, 1.375, 1.25, 1.375, 1.125, 1.375, 1.375, 1.125, 1.375, 1.375, 1.375, 1, 1.25, 1.125, 1.375, 1, 1.375, 1.125, 1, 1.25, 1.375, 1, 1.375, 1, 1, 1.125, 1.125, 1, 1.25, 1.25, 1.25, 1.125, 1.125, 1.125, 1.125, 1.25, 1.125, 1.375, 1.375, 1.25, 1, 1, 1.375, 1, 1.375, 1.125, 1, 1, 1.375, 1, 1.125, 1.25, 1.25, 1.375, 1.25, 1, 1, 1, 1.25, 1.125, 1.25, 1.375, 1.25, 1.375, 1.25, 1.25, 1.25, 1, 1.125, 1, 1, 1.125, 1, 1.375, 1.125, 1.375, 1.125, 1.125, 1.25, 1.125, 1, 1.25, 1, 1.375, 1.375, 1.125, 1.25, 1, 1, 1.375, 1, 1.375, 1.375, 1.25, 1.125$\}$

Grid 4 $G_s = \{2.17, 2, 1.5, 1.83, 2.17, 1.33, 1.67, 1.83, 2.17, 1.17, 1.33, 1.67, 1.5, 2, 1.33, 1.5, 1.33, 1.5, 1, 1.33, 1, 1, 1.5, 1.67, 2, 1.67, 1.83, 1.33, 1.33, 2.17, 1.33, 1, 1.33, 1.33, 2.17, 2, 1.33, 1.83, 1, 1.5, 1.33, 1.33, 1.83, 1.83, 2, 2, 1.67, 1.17, 1.33, 1.5, 1.67, 2.17, 2, 2.17, 2.17, 1.17, 1.67, 2, 2.17, 2, 1.33, 1.83, 1.83, 1.83, 1.83, 1.33, 2.17, 1.33, 1.33, 1.67, 1, 1.5, 2, 1.5, 1, 1.33, 2.17, 2, 2, 1.33, 1.17, 2, 2, 1.17, 2.17, 1.5, 1.17, 1.67, 1.5, 1.67, 1.67, 1.5, 1.33, 1.67, 2, 1.83, 2.17, 1.33, 1, 2, 2.17, 1.67, 1.67\}$

Grid 5 $G_s = \{5, 1, 7.5, 3, 5, 7, 7.5, 4.5, 3.5, 6, 4.5, 2, 6.5, 1.5, 2.5, 1, 6.5, 2.5, 1.5, 1, 7.5, 5, 3, 1, 7, 3.5, 2.5, 5.5, 1, 3.5, 3.5, 4, 1.5, 5, 2, 1, 6, 7.5, 1, 8, 7.5, 6, 2, 8, 1.5, 8, 3.5, 8, 2, 5, 2.5, 5, 2, 7.5, 5.5, 4, 8, 3.5, 8.5, 2.5, 3, 3, 7, 1, 6.5, 6, 8.5, 8, 3.5, 1, 3, 3, 2.5, 3.5, 2, 6.5, 8.5, 8.5, 3.5, 2.5, 3, 2.5, 7, 1, 6, 7, 7, 5, 2, 5, 1.5, 6, 4, 7.5, 3.5, 6.5, 7, 5.5, 1, 2.5, 4, 4, 5, 1, 7, 8.5, 5, 8, 1, 2.5, 2, 3.5, 1, 1, 3, 6\}$

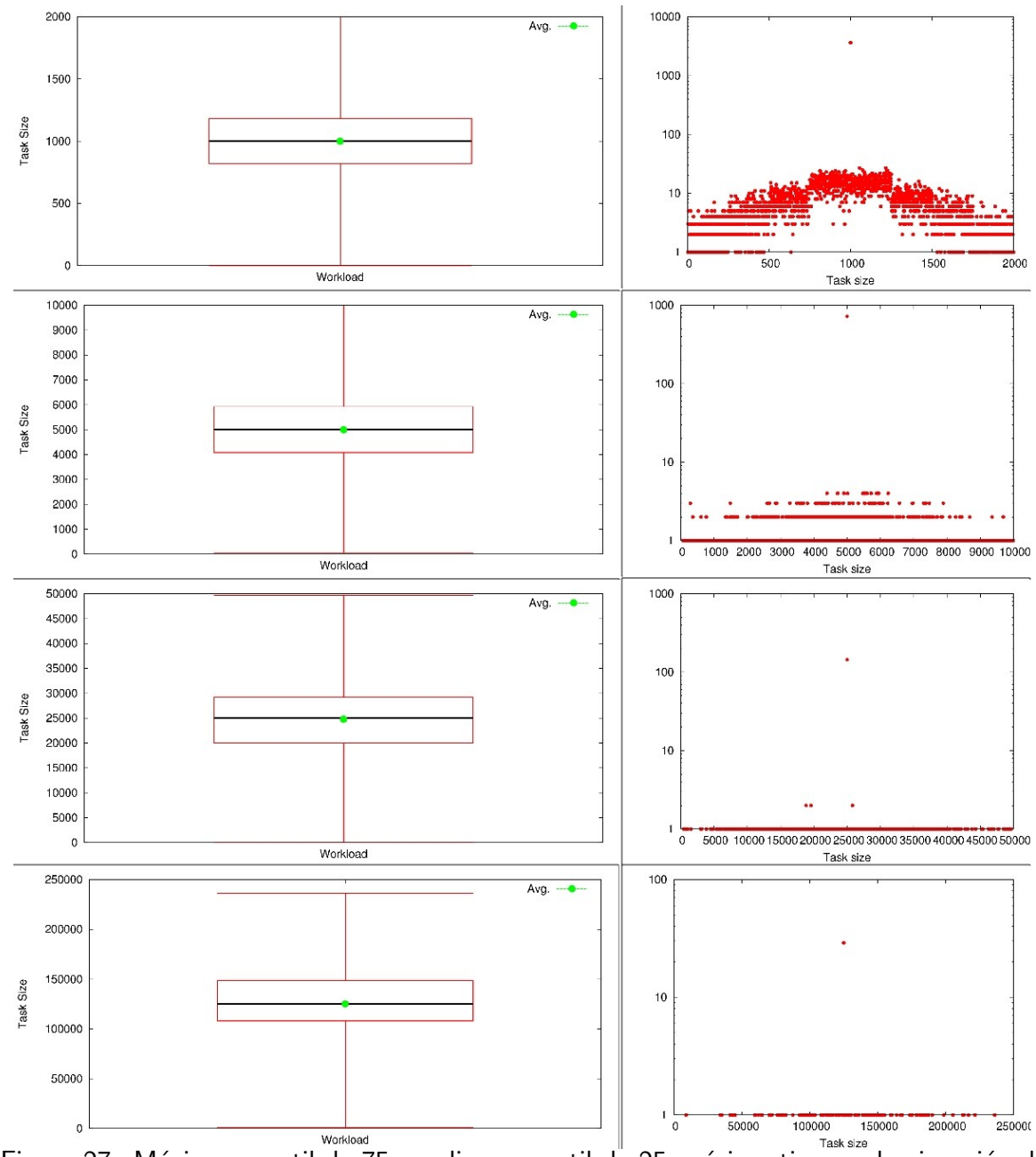

Figura 27. Máximo, cuartil de 75, mediana, cuartil de 25, mínimo tiempo de ejecución de los grupos, así como histogramas de $\bar{p_j} = \{1000, 5000, 25000, 125000\}$.

Apéndice B

Configuración para los experimentos WQR-x

En este apéndice se presentan las distintas configuraciones de grid y cargas de trabajo para los experimentos presentados en el capítulo 5.

Grid 1 $G_s = \{$1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1$\}$

Grid 1 $G_{eff} = \{$1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1$\}$

Grid 2 $G_s = \{$1, 1, 1.11, 1, 1, 1, 1.11, 1.11, 1.11, 1, 1, 1.11, 1.11, 1, 1.11, 1.11, 1.11, 1.11, 1.11, 1.11, 1.11, 1.11, 1, 1, 1.11, 1.11, 1.11, 1.11, 1.11, 1.11, 1, 1, 1, 1.11, 1, 1.11, 1.11, 1.11, 1, 1, 1, 1, 1.11, 1.11, 1.11, 1, 1.11, 1, 1.11, 1.11, 1.11, 1, 1.11, 1.11, 1, 1, 1.11, 1.11, 1.11, 1, 1, 1, 1, 1.11, 1.11, 1.11, 1, 1, 1.11, 1, 1, 1, 1.11, 1, 1, 1.11, 1, 1, 1.11, 1.11, 1, 1.11, 1, 1, 1.11, 1, 1.11, 1, 1, 1.11, 1.11, 1.11, 1, 1.11, 1, 1, 1, 1.11, 1, 1, 1, 1.11, 1, 1.11, 1, 1.11$\}$

Grid 2 $G_{eff} = \{$1.11, 1.11, 1, 1.11, 1, 1.11, 1, 1, 1.11, 1.11, 1, 1, 1, 1.11, 1.11, 1.11, 1, 1, 1.11, 1, 1.11, 1, 1, 1, 1, 1, 1, 1.11, 1.11, 1.11, 1.11, 1, 1, 1.11, 1.11, 1.11, 1, 1.11, 1, 1, 1, 1.11, 1, 1, 1, 1, 1.11, 1.11, 1.11, 1, 1, 1, 1.11, 1.11, 1, 1, 1.11, 1, 1.11, 1, 1,

1.11, 1.11, 1, 1, 1, 1.11, 1.11, 1.11, 1, 1.11, 1.11, 1.11, 1, 1, 1.11, 1.11, 1, 1, 1.11, 1, 1, 1, 1.11, 1.11, 1, 1, 1, 1.11, 1.11, 1.11, 1.11, 1.11, 1.11, 1, 1.11, 1, 1, 1, 1.11, 1.11, 1.11, 1.11, 1.11, 1}

Grid 3 G_s = {1, 1.25, 1, 1, 1.25, 1.25, 1.125, 1.125, 1, 1.375, 1.375, 1.125, 1.375, 1.25, 1.375, 1.125, 1.375, 1.375, 1.125, 1.375, 1.375, 1.375, 1, 1.25, 1.125, 1.375, 1, 1.375, 1.125, 1, 1.25, 1.375, 1, 1.375, 1, 1, 1.125, 1.125, 1, 1.25, 1.25, 1.25, 1.125, 1.125, 1.125, 1.125, 1.25, 1.125, 1.375, 1.375, 1.25, 1, 1, 1.375, 1, 1.375, 1.125, 1, 1, 1.375, 1, 1.125, 1.25, 1.25, 1.375, 1.25, 1, 1, 1, 1.25, 1.125, 1.25, 1.375, 1.25, 1.375, 1.25, 1.25, 1.25, 1, 1.125, 1, 1, 1.125, 1, 1.375, 1.125, 1.375, 1.125, 1.125, 1.25, 1.125, 1, 1.25, 1, 1.375, 1.375, 1.125, 1.25, 1, 1, 1.375, 1, 1.375, 1.375, 1.25, 1.125}

Grid 3 G_{eff} = {1, 1, 1.25, 1.375, 1.375, 1.25, 1, 1.375, 1, 1.125, 1.125, 1.25, 1.25, 1.25, 1.375, 1.125, 1.125, 1.125, 1.375, 1.375, 1.25, 1.125, 1.125, 1.375, 1.375, 1.25, 1.25, 1.125, 1.375, 1, 1.125, 1.25, 1.375, 1.375, 1.25, 1.125, 1, 1.25, 1.25, 1.125, 1.375, 1, 1, 1, 1.125, 1.375, 1.125, 1.25, 1.25, 1.125, 1.125, 1.375, 1.125, 1, 1.375, 1.125, 1.375, 1.125, 1.25, 1.25, 1.125, 1, 1.125, 1, 1.125, 1, 1.25, 1, 1, 1.375, 1.125, 1.375, 1.25, 1.125, 1.25, 1.25, 1, 1.25, 1.375, 1.25, 1.25, 1.125, 1.125, 1.375, 1, 1.125, 1.125, 1.375, 1.375, 1.375, 1.375, 1.375, 1.25, 1.25, 1, 1.25, 1.25, 1, 1.375, 1.125, 1.375, 1.25, 1.125, 1.125, 1.375, 1.125}

Grid 4 G_s = {2.17, 2, 1.5, 1.83, 2.17, 1.33, 1.67, 1.83, 2.17, 1.17, 1.33, 1.67, 1.5, 2, 1.33, 1.5, 1.33, 1.5, 1, 1.33, 1, 1, 1.5, 1.67, 2, 1.67, 1.83, 1.33, 1.33, 2.17, 1.33, 1, 1.33, 1.33, 2.17, 2, 1.33, 1.83, 1, 1.5, 1.33, 1.33, 1.83, 1.83, 2, 2, 1.67, 1.17, 1.33, 1.5, 1.67, 2.17, 2, 2.17, 2.17, 1.17, 1.67, 2, 2.17, 2, 1.33, 1.83, 1.83, 1.83, 1.83, 1.33, 2.17, 1.33, 1.33, 1.67, 1, 1.5, 2, 1.5, 1, 1.33, 2.17, 2, 2, 1.33, 1.17, 2, 2, 1.17, 2.17, 1.5, 1.17, 1.67, 1.5, 1.67, 1.67, 1.5, 1.33, 1.67, 2, 1.83, 2.17, 1.33, 1, 2, 2.17, 1.67,

1.67}

Grid 4 G_{eff} = {1.5, 1.17, 1.83, 1, 1.67, 1.33, 2.17, 2, 1.83, 2.17, 1.83, 1.17, 1.67, 1, 1.33, 1.33, 2, 1, 1.33, 2, 2.17, 1.33, 1.33, 2, 1.5, 2, 1.17, 2, 2, 2, 1.5, 1.17, 1.33, 2, 1.83, 2, 1, 2.17, 2, 1, 2, 2.17, 1.67, 2, 1.67, 1.5, 2, 1.17, 1.33, 1, 1.5, 1.83, 1.33, 1.5, 1.5, 2, 1.5, 2, 1.5, 1.17, 2.17, 1.83, 1.33, 1, 1.33, 2.17, 1.17, 1.17, 1.83, 1.33, 1, 1, 1.83, 1.33, 1.5, 1, 1, 1.83, 2.17, 1.83, 1.67, 1.5, 1.83, 1.33, 2, 1.67, 1, 2, 1, 1.83, 1.67, 1.67, 1.83, 2.17, 2.17, 1.83, 1.67, 1.5, 2, 2.17, 1.17, 1, 1.17}

Grid 5 G_s = {5, 1, 7.5, 3, 5, 7, 7.5, 4.5, 3.5, 6, 4.5, 2, 6.5, 1.5, 2.5, 1, 6.5, 2.5, 1.5, 1, 7.5, 5, 3, 1, 7, 3.5, 2.5, 5.5, 1, 3.5, 3.5, 4, 1.5, 5, 2, 1, 6, 7.5, 1, 8, 7.5, 6, 2, 8, 1.5, 8, 3.5, 8, 2, 5, 2.5, 5, 2, 7.5, 5.5, 4, 8, 3.5, 8.5, 2.5, 3, 3, 7, 1, 6.5, 6, 8.5, 8, 3.5, 1, 3, 3, 2.5, 3.5, 2, 6.5, 8.5, 8.5, 3.5, 2.5, 3, 2.5, 7, 1, 6, 7, 7, 5, 2, 5, 1.5, 6, 4, 7.5, 3.5, 6.5, 7, 5.5, 1, 2.5, 4, 4, 5, 1, 7, 8.5, 5, 8, 1, 2.5, 2, 3.5, 1, 1, 3, 6}

Grid 5 G_{eff} = {5, 6.5, 4, 4.5, 7, 2, 5.5, 3, 5.5, 3.5, 7.5, 7.5, 3.5, 3.5, 2, 2.5, 6, 2, 1.5, 7.5, 1.5, 4, 3.5, 7.5, 8, 8, 3, 5, 7.5, 7, 4, 3, 7.5, 2, 1.5, 8.5, 8, 2, 1, 4, 2.5, 3, 4, 3, 5.5, 3, 5, 6.5, 5.5, 8, 4, 7, 2, 4, 3, 4.5, 5, 4, 2.5, 2.5, 1.5, 6.5, 7.5, 2, 2, 6, 8, 5, 6, 2, 1.5, 3.5, 7.5, 1.5, 6, 5, 2, 8.5, 5.5, 3, 5, 8.5, 2, 6, 6.5, 2.5, 3.5, 3, 6, 8, 5, 3.5, 6, 4.5, 6.5, 3.5, 3, 8.5, 3.5, 4.5, 5, 7, 8.5, 5.5, 1, 8, 1, 6, 4.5, 6, 4, 3, 2, 7, 5, 2}

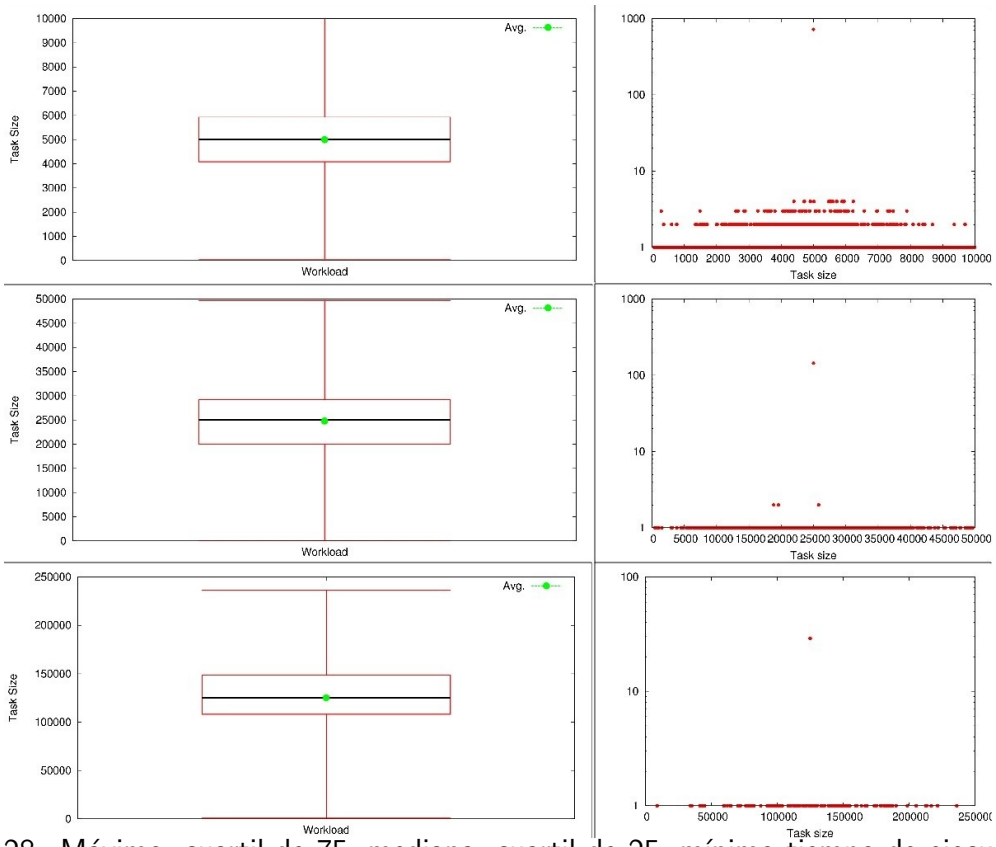

Figura 28. Máximo, cuartil de 75, mediana, cuartil de 25, mínimo tiempo de ejecución de los grupos, 09 como histogramas de $\bar{p_j} = \{5000, 25000, 125000\}$.