Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California

Maestría en Ciencias
en Ciencias de la Computación

Estrategias de optimización para la gestión de recursos en nubes basadas en contenedores

Tesis
para cubrir parcialmente los requisitos necesarios para obtener el grado de Maestro en Ciencias

Presenta:

Rewer Miguel Canosa Reyes

Ensenada, Baja California, México
2018

Tesis defendida por

Rewer Miguel Canosa Reyes

y aprobada por el siguiente Comité

Dr. Andrey Chernykh
Director de tesis

Miembros del comité

Dr. Carlos Alberto Brizuela Rodríguez

Dr. Raúl Rivera Rodríguez

Dr. Ubaldo Ruiz López

Dr. Jesús Favela Vara
Coordinador del Posgrado en Ciencias de la Computación

Dra. Rufina Hernández Martínez
Directora de Estudios de Posgrado

Resumen de la tesis que presenta **Rewer Miguel Canosa Reyes** como requisito parcial para la obtención del grado de Maestro en Ciencias en Ciencias en de la Computación.

Estrategias de optimización para la gestión de recursos en nubes basadas en contenedores

Resumen aprobado por:

_____

Dr. Andrey Chernykh

Director de tesis

La computación en la nube es un paradigma para el alojamiento y la prestación de servicios a través de Internet. Muchas empresas y gobiernos han comenzado a utilizarlo debido a las ventajas que ofrece. Una tecnología base del cómputo en la nube es la virtualización, que utiliza estructuras como máquinas virtuales y contenedores. Los contenedores permiten la portabilidad de aplicaciones y entornos con imágenes de sistema operativo de menor tamaño que en máquinas virtuales. Sin embargo, los centros de datos de la nube son altos consumidores de energía debido a la gran cantidad de equipamiento que los conforman. Por otra parte, en la contratación de los servicios de cómputo se definen los acuerdos a nivel de servicio (SLA) que establecen los compromisos de los proveedores de servicio de la nube con los clientes en el arrendamiento de recursos computacionales. En el presente trabajo se propone un conjunto de estrategias de asignación de trabajos a contenedores, y se evalúa su impacto en la disminución de las violaciones SLA, el tiempo de terminación de todos los trabajos y en el consumo de energía en centros de datos de la nube basados en contenedores. Se estudiaron dos escenarios relacionados con la capacidad de procesamiento que se puede asignar a las tareas. Para evaluar el desempeño de las estrategias propuestas se utiliza el simulador CloudSim. Las estrategias del segundo escenario reducen el tiempo de terminación de todos los trabajos y las violaciones de SLA sin incrementar significativamente el consumo de energía.

Palabras clave: cómputo en la nube, contenedores, estrategias de calendarización.

Abstract of the thesis presented by **Rewer Miguel Canosa Reyes** as a partial requirement to obtain the Master of Science degree in Computer Science.

Resource optimization strategies in containerized clouds

Abstract approved by:

_____

Dr. Andrey Chernykh
Thesis Director

Cloud computing is a paradigm for the hosting and provision of services through the Internet. Many companies and governments have started to use it because of the advantages it offers. A core technology of cloud computing is virtualization, which uses structures such as virtual machines and containers. The containers allow the portability of applications and environments with operating system images of smaller size than in virtual machines. However, cloud data centers are high energy consumers due to the large amount of equipment that make them up. On the other hand, in the contracting of computing services, service-level agreements (SLA) providing the obligations of cloud service providers (CSP) towards customers in the leasing of computational resources are defined. This work proposes a set of strategies for assigning jobs to containers and assesses their impact on the reduction of SLA violations, the completion time of all jobs and energy consumption in container-based cloud data centers. Two scenarios related to the processing capacity that can be assigned to tasks were studied. To evaluate the performance of the proposed strategies, the CloudSim simulator is used. The strategies of the second scenario reduce the completion time of all works and SLA violations without significantly increasing energy consumption.

Keywords: cloud computing, containers, scheduling strategies.

Dedicatoria

A Dios por la fuerza, por poner las personas adecuadas cuando todo parecía más difícil.

A mi eterna Rosalina, fuente de toda mi inspiración, luz y faro de mi vida. Por tallar en mi alma los valores que me han convertido en el hombre que soy.

A mi familia gracias por todo el apoyo y amor incondicional que siempre me han brindado.

A mis compañeros de generación, por ser tan diversos y tan únicos, por enseñarme tanto y por apoyarme más.

A mis amigos de siempre, los que nunca fallan sin importar cuan difíciles sean los tiempos.

A todas las personas que hacen posible la investigación en este hermoso país que es México, a su gente trabajadora y cordial, a su pueblo alegre que por medio del Consejo Nacional de Ciencia y Tecnología (CONACyT) me otorgaron una beca para la realización de este trabajo de investigación.

A la Cuba que llevo dentro, que siempre me acompaña y que no ha de salir nunca de mí.

Agradecimientos

A mi director de tesis, el Dr. Andrey Chernykh por su constante orientación, su apoyo incondicional y su confianza en mí.

A los miembros del comité de tesis, el Dr. Carlos Alberto Brizuela Rodríguez, el Dr. Ubaldo Ruiz y el Dr. Raúl Rivera Rodríguez, por su generosa orientación, sus comentarios constructivos, correcciones de tesis y su retroalimentación sobre el trabajo de investigación.

A los doctores Fermín Alberto Armenta Cano y Jorge Mario Cortés Mendoza quienes me apoyaron incondicionalmente y fueron mi sostén por mucho tiempo.

Al Dr. Eduardo René Concepción Morales por estar siempre presente en mi formación y mostrarme el valor del conocimiento con su ejemplo.

A mis compañeras y compañeros del cubo, por mostrar dedicación ante el trabajo y ser ejemplo para mí.

A todos los estudiantes y personal de CICESE, en especial a los miembros del departamento de ciencias de la computación por su apoyo y amistad.

Al Centro de Investigación Científica y de Educación Superior de Ensenada (CICESE), excelente institución que me acogió para realizar mis estudios de posgrado, por proporcionar las instalaciones y los medios necesarios para realizar mi investigación.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por brindar el apoyo económico para realizar mis estudios de maestría.

Tabla de contenido

Página

Lista de figuras

Lista de tablas

Capítulo 1. Introducción

1.1 Cómputo en la nube

En los últimos años se ha dado un notable avance en las tecnologías de la información y las comunicaciones. Debido a un mayor acceso al Internet, el abaratamiento de los recursos de cómputo y la necesidad creciente de disponer de manera ubicua de hardware más potente, se han desarrollado nuevas formas de brindar y acceder a recursos de cómputo.

Esta tendencia tecnológica ha permitido la construcción de un nuevo modelo de acceso a los recursos computacionales llamado Cómputo en la Nube (Cloud Computing). El Instituto Nacional de Tecnologías y Estándares de Estados Unidos (NIST por sus siglas en inglés) en (Mell and Grance 2011) define al Cómputo en la Nube como "un modelo para permitir de forma conveniente y por medio de la red, el acceso a un conjunto compartido de recursos informáticos, que se pueden configurar bajo demanda" (por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser rápidamente provistos y liberados con un esfuerzo mínimo de gestión o interacción por parte del proveedor de servicios de la nube(Cloud Service Provider - CSP por sus siglas en inglés).

En (Ahmad 2017) se muestra la evolución de la computación en la nube. Esta se remonta al concepto de virtualización y el de no interferencia, propuesto en los años ochenta. La virtualización, en el contexto de la computación en la nube, se refiere a albergar una vista unificada y uniforme de los distintos recursos de hardware para el sistema huésped y el software de aplicación.

El cómputo en la nube ha sido adoptado por empresas y gobiernos como se detalla en (Pereira, Da Silva, Batista, Delicato, Pires, and Khan 2017), donde se destacan los beneficios del manejo de los recursos de tecnologías de la información tanto para el sector gubernamental como privado.

La combinación de las potencialidades del cómputo en la nube tales como: el acceso a los recursos de cómputo, así como el acceso a las aplicaciones desde cualquier dispositivo y la reducción en los costos, han favorecido la entrega de estos servicios agrupados generalmente en: Plataforma como Servicio (PaaS – Plataform as a Service), Software como Servicio (SaaS – Software as a Service), Infraestructura como Servicio (IaaS - Infrastructure as a Service). Las particularidades de estos modelos de entrega de servicio se describen en (Rastogi and Sushil 2015).

PaaS es una categoría de servicios en la nube que proporciona una plataforma y un entorno que permiten a los desarrolladores crear aplicaciones y servicios que funcionen a través de Internet.

SaaS se describe como cualquier servicio en la nube, en el que los consumidores puedan acceder a las aplicaciones de software a través de Internet. Estas aplicaciones están alojadas en la nube y pueden utilizarse para una amplia variedad de tareas, tanto para organizaciones particulares como para públicas. Las aplicaciones como Google mail, Twitter, Facebook y Flickr son ejemplos de SaaS, en las cuales los usuarios pueden acceder a los servicios a través de cualquier dispositivo que pueda conectarse a Internet.

IaaS proporciona acceso a recursos informáticos situados en un entorno virtualizado, a través de Internet. En este caso, los recursos informáticos ofrecidos consisten en la virtualización de hardware como infraestructura de procesamiento. La definición de IaaS abarca aspectos como el espacio en servidores virtuales, conexiones de red, ancho de banda, direcciones IP y balanceadores de carga (Kokane, Jain, and Sarangdhar 2013).

Empresas como Google y Amazon, han incorporado un nuevo tipo de servicio, llamado Contenedores como Servicio (CaaS - Container as a Service), el cual se describe en (Wang and Wu 2009) como: una forma de servicios para la virtualización basada en contenedores, en el que los proveedores de servicios de nube ofrecen un marco completo a los clientes para desplegar y gestionar contenedores, aplicaciones y clústeres.

Este tipo de servicio ayuda a los programadores y departamentos de TI a desarrollar, ejecutar y administrar aplicaciones. Sin embargo, se asocia más como un subconjunto de IaaS. El recurso básico para CaaS es un contenedor (ver subepígrafe 2.1), en lugar de una máquina virtual (VM) o un sistema de hardware, que se utilizan para soportar ambientes IaaS. Sin embargo, el contenedor puede ejecutarse dentro de una VM o interactuar directamente con el hardware físico contratado.

Los contenedores pueden considerarse una revolución en la era de la nube, ya que son ligeros, fáciles de configurar, gestionar y pueden reducir considerablemente el tiempo de puesta en marcha. En la Figura 1 se muestra el modelo CaaS que utiliza Docker, el cual es un sistema de gestión de contenedores. Es importante notar que el modelo CaaS se encuentra entre los modelos IaaS y PaaS (Piraghaj, Dastjerdi, Calheiros, and Buyya 2015).

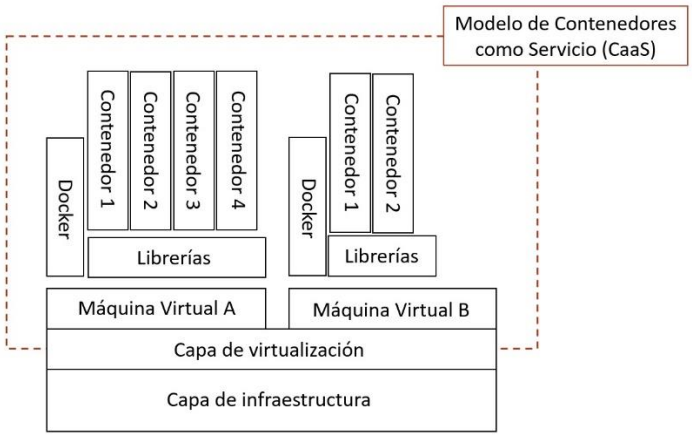

Figura 1. Modelo contenedor como servicio.

La capacidad de adquirir y liberar recursos bajo demanda es una de las características clave de la computación en la nube. Un elemento importante para un CSP es la calidad del servicio (QoS - Quality of Service). El objetivo de un CSP, en este caso, es asignar y remover recursos de la nube para satisfacer sus objetivos de nivel de servicio (SLO), como se señala en (Endo, Batista, Goncalves, Rodrigues, Sadok, Kelner, Sefidcon, and Wuhib 2013); no resulta fácil determinar SLOs como requisitos de QoS asociados a necesidades de recursos de bajo nivel, como CPU y requisitos de memoria principal.

Dado que los usuarios realizan la contratación de servicios asociados a recursos de cómputo, es necesario que éstos tengan garantía sobre el servicio que están contratando. Como se señala en (Garg, Gopalaiyengar, and Buyya 2011), estas garantías, que se documentan en forma de un Acuerdo de Nivel de Servicio (SLA), son cruciales ya que sólo entonces los clientes pueden confiar en la externalización de sus trabajos a las nubes.

Los acuerdos de nivel de servicio representan un contrato que especifica las obligaciones mínimas del proveedor hacia sus clientes o lo que esperan recibir los clientes a cambio del precio pagado. El propósito primario de un SLA es especificar las expectativas del funcionamiento, establecer la responsabilidad, y detallar las alternativas y consecuencias si el funcionamiento o la calidad del servicio (QoS) no son los acordados por ambas partes. El SLA es un factor crítico para el éxito en la contratación de servicios proporcionados por un tercero, por este motivo es la base para la entrega de servicios de cómputo en la nube (Alhamad, Dillon, and Chang 2010b).

Existen cuatro modelos fundamentales para la implementación de la computación en la nube: público, privado, híbrido y comunitario. Éstos delimitan en qué forma se puede acceder a los servicios de la nube.

Algunas características esenciales que identifican al paradigma de cómputo en la nube se señalan en (Rastogi and Sushil 2015) y son:

- Agrupamiento de recursos: los recursos computacionales del proveedor de servicios se agrupan para servir a múltiples consumidores que usan el modelo de múltiples arrendatarios.
- Amplio acceso desde la red: las capacidades de la nube están disponibles a través de la red y es posible acceder a estas a través de cualquier cliente.
- Rápida elasticidad: los recursos de hardware y software requeridos pueden adaptarse con facilidad en dependencia de las necesidades del cliente.
- Servicios a la medida: los servicios de computación en la nube utilizan capacidades de medición que permiten controlar y optimizar el uso de recursos; *autoservicio bajo demanda*: todos los servicios son gestionados sin la necesidad de interacción humana.

1.1.1 Máquinas Virtuales

La virtualización permite combinar o dividir recursos computacionales físicos (hardware) para suministrarlos como ambientes funcionales para la ejecución de sistemas operativos y aplicaciones (software). Este proceso es transparente para los usuarios que utilizan entornos virtuales. Los inicios de la virtualización se remontan a la década del 60 por parte de IBM (Semnanian, Pham, Englert, and Wu 2010).

Una de las tecnologías base del paradigma de cómputo en la nube es la virtualización (Zhao, Mandagere, Alatorre, Mohamed, and Ludwig 2015). La virtualización de recursos computacionales es uno de los conceptos claves para el cómputo en la nube incluyendo sistema operativo, memoria, almacenamiento u otros recursos de redes (Celesti, Mulfari, Fazio, Villari, and Puliafito 2016). La capa de virtualización puede manejar máquinas virtuales o contenedores.

Una máquina virtual (VM) es, en esencia, la abstracción de un servidor físico que dispone de un sistema operativo, procesador, memoria, almacenamiento y recursos de red. Una VM ejecuta programas como si fuera una computadora real. En un ordenador o servidor físico se pueden ejecutar al mismo tiempo varios sistemas operativos y aplicaciones, al colocar en él varias máquinas virtuales (Figura 2). La capa de software, conocida como hipervisor, desvincula las máquinas virtuales de la máquina física y asigna dinámicamente los recursos a cada ordenador virtual según las necesidades.

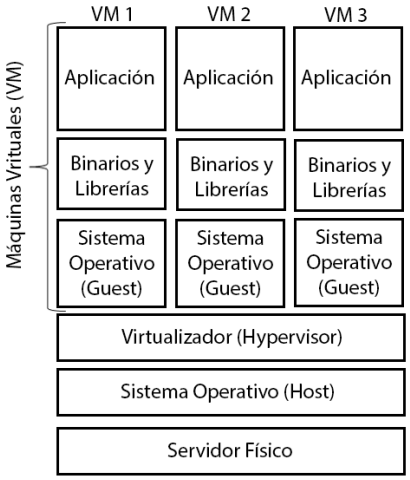

Figura 2. Implementación de VM en servidor físico.

La utilización de máquinas virtuales ofrece beneficios tales como: si se desconfigura un servidor o un sistema operativo virtualizado, es sumamente fácil de restaurar en comparación con un servidor físico.

La utilización del equipamiento disponible es más eficiente, debido a que un mismo servidor físico puede ejecutar varios sistemas operativos con multitud de servicios de manera aislada, disminuyendo así la cantidad de equipos físicos para realizar las mismas tareas. Esto reduce el consumo de energía y disminuyen los costos de mantenimiento a los equipos. Se realiza una gestión más eficiente del hardware utilizando balanceo de carga y disminuyendo los tiempos de rehabilitación de los servicios ante una posible avería; los recursos de cómputo pueden ampliarse de forma dinámica en un entorno productivo.

El empleo de esta tecnología trae riesgos asociados: se necesitan servidores con buenas prestaciones, un daño en el sistema operativo anfitrión puede ocasionar que las máquinas virtuales asociadas a éste dejen de funcionar, entre otros.

1.1.2 Contenedores

Un contenedor (CT) es un concepto clave de virtualización que permite el encapsulamiento de aplicaciones basados en una estructura que se comunica con un sistema operativo (Figura 3). Una característica que permite aprovechar de manera eficiente los recursos computacionales, es que varios contenedores pueden intercambiar llamadas al mismo sistema operativo base sin requerir una instancia de este por cada contenedor como sucede con las máquinas virtuales.

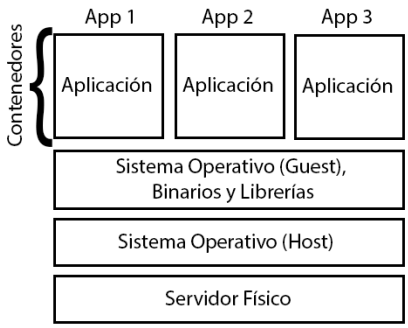

Figura 3. Implementación de CT en servidor físico.

El empleo de esta tecnología ofrece beneficios entre los que se destacan: instalación más sencilla, ya que éstos se inician a partir de imágenes preestablecidas que incluyen componentes requeridos como librerías y archivos de configuración, por lo general su instalación se reduce a la introducción de una línea de comandos; independencia de la plataforma: las imágenes de los CTs se pueden transportar cómodamente de un sistema a otro sin afectar su funcionamiento; aplicaciones aisladas: cada programa funciona independientemente de otros contenedores, de forma que aplicaciones con requerimientos opuestos pueden funcionar en paralelo en el mismo sistema; gestión única de recursos: la asignación dinámica de recursos de hardware a los contenedores se realiza mediante una interfaz única para su gestión y aprovisionamiento.

Uno de los elementos a destacar en este aspecto es la seguridad, debido a que varios contenedores realizan llamadas al mismo sistema operativo base, como los contenedores se instalan a partir de imágenes, es posible que se introduzcan código malicioso por sus creadores, lo que puede comprometer el desempeño de la infraestructura cuando el contenedor infectado se encuentre en ejecución en un ambiente productivo.

1.1.3 Máquinas Virtuales vs Contenedores

En (Dua, Raja, and Kakadia 2014) se realiza una comparativa entre los dos tipos de instancias de virtualización que se emplean en las nubes de cómputo: máquinas virtuales y contenedores. La Tabla 1 presenta los resultados de la comparativa, donde se aprecian los aspectos que favorecen el empleo de contenedores sobre la utilización de máquinas virtuales.

Tabla 1. Comparativa de características entre máquinas virtuales y contenedores.

Parámetro	Máquina Virtual	Contenedor
Sistema operativo	Cada máquina virtual se ejecuta en un hardware virtual y el núcleo (kernel) es cargado en su propia región de memoria.	Todos los contenedores comparten el mismo sistema operativo y kernel. La imagen del kernel es cargada en la memoria física.
Comunicación	A través de los dispositivos Ethernet.	Mecanismos IPC estándar como señales, tuberías, sockets, etc.
Seguridad	Depende de la implementación del hipervisor.	Se puede aprovechar el control de acceso obligatorio.
Rendimiento	Las máquinas virtuales sufren de una pequeña sobrecarga, ya que las instrucciones de la máquina se traducen del sistema operativo huésped (guest) al sistema base (Host OS).	Los contenedores proporcionan un rendimiento casi nativo en comparación con el sistema base (Host OS) host subyacente.
Aislamiento	No es posible compartir bibliotecas, archivos, entre los sistemas operativos de éstas.	Los subdirectorios se pueden montar de forma transparente y se pueden compartir.
Tiempo de inicio	Tardan unos minutos en arrancar.	Pueden arrancar en unos pocos segundos en comparación con las máquinas virtuales.
Almacenamiento virtual	Requieren mucho más almacenamiento, ya que todo el kernel del sistema operativo y sus programas asociados deben instalarse y ejecutarse.	Toma una menor cantidad de almacenamiento ya que el sistema operativo base es compartido.

1.2 Calendarización en la nube

La asignación de recursos es una característica en constante evolución de muchos problemas vinculados a la administración de centros de datos y computación en la nube. Como se disponen de cuantiosos recursos de hardware y software, en este paradigma es vital hacer una gestión eficiente de los mismos.

La gestión de recursos en la computación en la nube se entiende como el proceso de asignar recursos de cómputo, almacenamiento y redes a un conjunto de aplicaciones para satisfacer sus requerimientos. En este entorno, la virtualización juega un papel fundamental en especial en el modelo de Infraestructura como servicio (IaaS) (Bansal, Gupta, and Kaur 2016).

En (Pandit, Chattopadhyay, Chattopadhyay, and Chaki 2014) se precisa, como uno de los principales problemas, la asignación de cargas de trabajos a recursos con alta utilización. Los autores presentan un algoritmo de asignación de recursos utilizando un algoritmo de búsqueda llamado recocido simulado; se realizan experimentos para ilustrar la eficacia del algoritmo respecto al método de asignación comúnmente utilizado *El primero que llega es el primero que se asigna* (First-Come First-Serve).

Se pueden diferenciar varios tipos de calendarización en la nube, entre los que se encuentran la calendarización dinámica, estática, con prioridades, de flujos de trabajo, en tiempo real, heurística y de nivel de usuario. Una descripción más completa de estos tipos de calendarización, y de los métodos utilizados, se describen en (Tikar, Jaybhaye, and Pathak 2016).

En (Tikar, Jaybhaye, and Pathak 2016), se presenta un estudio comparativo del desempeño de varios algoritmos en el contexto de la computación en la nube, tomando en cuenta aspectos tales como: la viabilidad; su adecuación; su adaptabilidad; los parámetros de los algoritmos; entre otros.

El trabajo de (Zhao, Mohamed, and Ludwig 2018), aborda el problema de calendarización de contenedores y propone un modelo que respeta, tanto el balance de carga de trabajo, como el desempeño de las aplicaciones. Los resultados experimentales muestran que el sistema propuesto incrementa significativamente el desempeño de las aplicaciones preservando un alto balance de carga de trabajo.

1.3 Justificación

El cómputo en la nube utiliza técnicas de virtualización para lograr la elasticidad de los recursos compartidos a gran escala. Las máquinas virtuales por lo general son la columna vertebral en la capa de infraestructura. El empleo de contenedores permite un mejor aprovechamiento de los recursos físicos del hardware donde se ejecutan. Los contenedores se crean en función de la necesidad de ejecutar aplicaciones en un entorno PaaS (Pahl, Brogi, Soldani, and Jamshidi 2017).

En este ámbito, resulta necesario analizar la asignación de trabajos en nubes basadas en contenedores, estudiando técnicas que propicien una mejor utilización de los recursos, así como una gestión eficiente de su consumo energético.

La presente tesis contribuye en el campo de la calendarización de tareas en nubes basadas en contenedores. El objetivo de la misma consiste en proponer un método de calendarización que busca la reducción del tiempo de terminación de todas las tareas, las violaciones SLA y el consumo energético de la infraestructura subyacente para ejecutar las tareas. Las contribuciones fundamentales de este trabajo son:

- Abordar el problema de la asignación de tareas en centros de datos basado en contenedores.
- Proponer estrategias de optimización para la minimización de violaciones de SLA, uso eficiente de la energía y reducción del tiempo de terminación de las tareas que se ejecutan en centros de datos.

- Implementar estrategias de optimización en nubes basadas en contenedores para mejorar la asignación de trabajos considerando la disminución del tiempo de terminación de las tareas, la energía utilizada para su procesamiento y la calidad del servicio.
- Realizar un análisis de las estrategias de optimización implementadas para minimizar las violaciones de SLA, el tiempo de terminación de todas las tareas y el consumo de energía de infraestructuras nube basadas en contenedores.

1.4 Objetivos

1.4.1 Objetivo general

Diseñar, implementar y analizar estrategias de asignación de tareas en línea para centros de datos de la nube, basados en tecnologías de contenedores, considerando como objetivos de optimización: el tiempo de terminación de todas las tareas, el consumo de energía y las violaciones a los acuerdos de nivel de servicio.

1.4.2 Objetivos específicos

- Conocer el estado del arte de las tecnologías de contenedores.
- Diseñar nuevas estrategias de asignación de tareas en contenedores para optimizar el tiempo de terminación de todas las tareas, el consumo de energía y violaciones SLA.
- Extender y modificar el simulador CloudSim para evaluar las estrategias propuestas.
- Validar las estrategias en el simulador CloudSim en diferentes escenarios.
- Comparar mediante un análisis experimental las estrategias del estado del arte y las propuestas en el presente trabajo.

1.5 Propuesta de solución

A continuación, se describe la metodología empleada para abordar el problema de diseño e implementación de estrategias.

Primeramente, se analizaron varios trabajos relacionados a la calendarización en la nube. La mayoría de estos trabajos emplean máquinas virtuales para proporcionar el servicio. Las máquinas virtuales se han utilizado con frecuencia debido a que bridan un entorno favorable para el aislamiento de

procesos, pero adicionan una sobrecarga en los recursos, tanto para su ejecución como para su monitoreo. También se revisaron trabajos enfocados a la asignación de tareas en contenedores para ambientes de cómputo en la nube que utilizan el modelo CaaS. Los contenedores poseen tiempos de arranque menores a las máquinas virtuales y pueden ejecutarse varios sobre el mismo sistema operativo base.

Con posterioridad, se estudiaron diversas estrategias de asignación donde se analiza el consumo de energía para la ejecución de cargas de trabajo en el cómputo en la nube. A partir del estudio realizado, se definieron las estrategias que constituyen el núcleo de la presente tesis.

Se implementaron en el simulador CloudSim (Calheiros, Ranjan, Beloglazov, De Rose, and Buyya 2011) las entidades necesarias para recrear el entorno que se considera en el problema de la presente tesis. Se realizaron los experimentos para validar las estrategias de asignación propuestas atendiendo a los niveles de carga del sistema y la necesidad de minimizar el tiempo de terminación de todas las tareas, las violaciones de SLA y el consumo energético de la infraestructura subyacente.

Se abordan dos escenarios para la distribución de la capacidad de procesamiento dentro del contenedor. En ambos escenarios, si la capacidad de procesamiento requerida por todos los trabajos es mayor a la del contenedor, la capacidad de procesamiento entregada a las tareas se disminuye de manera proporcional.

En el primer escenario, la máxima capacidad de procesamiento asignada a los trabajos está limitada por la capacidad definida en el SLA, el CSP minimiza el uso de la infraestructura para reducir el consumo de energía. Bajo este esquema, el contenedor puede tener capacidad de procesamiento disponible para su utilización en la ejecución de otros trabajos.

En el segundo escenario, el trabajo puede recibir hasta la totalidad de la capacidad de procesamiento del contenedor. El CSP maximiza el uso de la infraestructura para reducir el tiempo de ejecución de los trabajos. Una vez asignada la capacidad mínima de procesamiento a cada trabajo en el contenedor; se calcula la capacidad de procesamiento disponible y se asigna equitativamente entre todos los trabajos. La capacidad de procesamiento requerida por los trabajos siempre es menor o igual a la capacidad de procesamiento del contenedor.

A continuación, se expone un ejemplo por cada escenario para brindar al lector información detallada sobre la distribución de la capacidad de procesamiento.

La Figura 4 ilustra cómo se realiza la distribución de la capacidad de procesamiento del contenedor en el escenario 1, con capacidad máxima $q_1 = 1{,}000$ millones de instrucciones por segundo (MIPS). Para este ejemplo, han sido asignados los trabajos j_1, j_2 y j_3 al contenedor. El trabajo j_1 se ejecuta desde el instante 0 y demanda 300 MIPS de capacidad de procesamiento. Este trabajo, para completar su ejecución, debe ejecutar $3*10^{14}$ instrucciones. En el instante de tiempo $t = 50$ segundos, el trabajo j_2 se asigna al contenedor y comienza la ejecución de sus $2*10^{14}$ instrucciones demandando para ello 400 MIPS. En el segundo 70 comienza a ejecutarse el trabajo j_3. La capacidad de procesamiento requerida por este trabajo es de 500 MIPS y la cantidad de instrucciones a ejecutar para su terminación es $3.5*10^{14}$ instrucciones; en este instante ocurren 3 violaciones SLA debido a la reducción de la capacidad de procesamiento de todos los trabajos que se encuentran en ejecución. Esta reducción se debe a que las capacidades requeridas de 300, 400 y 500 MIPS exceden en 200 MIPS a q_1. A partir de este momento, se incumple en la entrega de capacidad de procesamiento a todos los trabajos por lo que se generan 3 violaciones de SLA.

En el instante de tiempo $t = 646$, el trabajo j_2 termina su ejecución por lo que se redistribuye la capacidad de procesamiento y se le asignan 300 MIPS a j_1 y 500 MIPS a j_3. En el tiempo $t = 866$ concluye la ejecución del trabajo j_3. Por último, para el instante de tiempo $t = 1096$, concluye su ejecución el trabajo j_1.

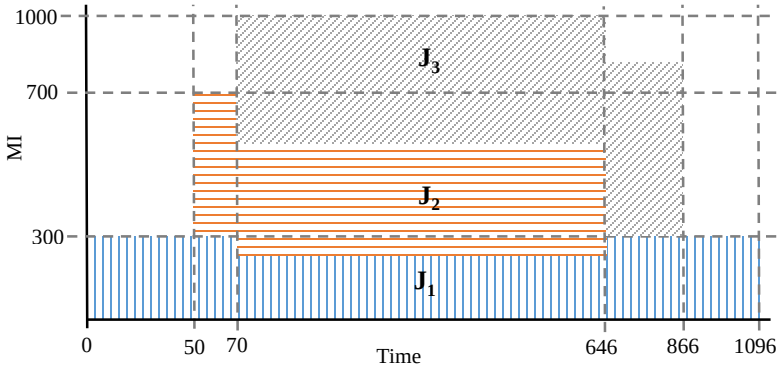

Figura 4. Distribución de la capacidad de procesamiento del contenedor para el escenario 1.

Utilizando los mismos trabajos que en el ejemplo anterior; la distribución de la capacidad de procesamiento del contenedor para el escenario 2 se puede apreciar en la Figura 5. El trabajo 1 se ejecuta desde el instante 0 y demanda 300 MIPS de capacidad de procesamiento, pero como no existen otros trabajos ejecutándose se le asignan los 1000 MIPS de los que dispone el contenedor. En el instante de tiempo $t = 50$, la capacidad de procesamiento de los trabajos 1 y 2 es de 450 y 550 MIPS, respectivamente. El trabajo j_3 comienza a ejecutarse en el segundo 70.

Al igual que en el ejemplo anterior se producen 3 violaciones SLA ya que las nuevas capacidades de procesamiento asignadas a los trabajos son de 249, 332 y 419 MIPS, respectivamente, todas estas capacidades son menores a las requeridas 300, 400 y 500 MIPS. En el instante de tiempo $t = 569$, concluye la ejecución del trabajo j_2. A partir de este instante se redistribuye nuevamente la capacidad de procesamiento siendo 400 MIPS para el trabajo j_1 y 600 MIPS para j_3.

En el tiempo $t = 804$, concluye la ejecución del trabajo 3 y nuevamente se asigna toda la capacidad de procesamiento del contenedor al trabajo j_1 que concluye 23 segundos más tarde.

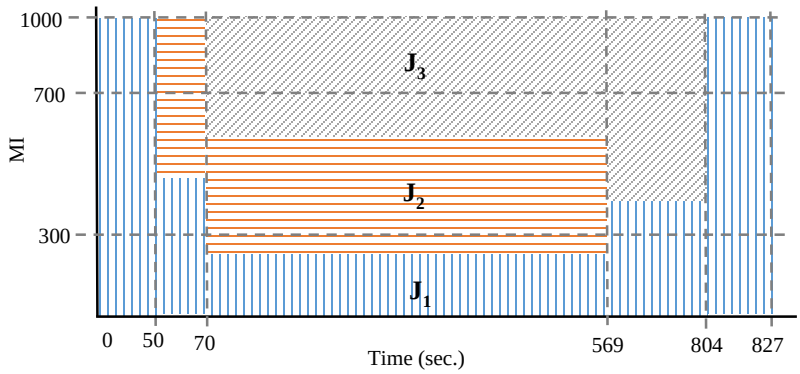

Figura 5. Distribución de la capacidad de procesamiento del contenedor para el escenario 2.

Nótese que en ambos escenarios ocurren 3 violaciones de SLA durante la ejecución de los trabajos. Sin embargo, para los escenarios mostrados existe una disminución del tiempo de terminación de todas las tareas de 269 segundos entre el primer y el segundo escenario.

En el primer escenario se persigue generar la menor cantidad de violaciones SLA, para ello solo se entrega a cada trabajo la capacidad requerida por este. Este elemento incide en un incremento del tiempo de terminación de todas las tareas, debido a que hay intervalos de tiempo donde existe capacidad de procesamiento del contenedor que no se utiliza para la ejecución de los trabajos.

En el segundo escenario, es posible que cada trabajo se ejecute con mayor capacidad de procesamiento de la que requiere, esto puede permitir una disminución en el tiempo de terminación de todas las tareas. La comparación entre los ejemplos anteriormente mostrados evidencia esta situación (ver Figuras 4 y 5).

Como parte de esta tesis se estudia el comportamiento de los dos escenarios mencionados con anterioridad. Sin embargo, puede decirse que existen muchos otros; por ejemplo, el que se muestra en la Figura 6. Luego del instante de tiempo $t = 70$, solamente se genera una violación SLA al afectar únicamente la capacidad de procesamiento requerida por el trabajo j_3. De esta forma, la capacidad de

procesamiento asignada a los trabajos j_1 y j_2 no se ve afectada logrando una pequeña reducción en tiempo de terminación de todas las tareas con respecto al escenario 1 pero aumentando considerablemente el tiempo con respecto al escenario 2. Esto también permite una disminución de 2 en las violaciones de SLA.

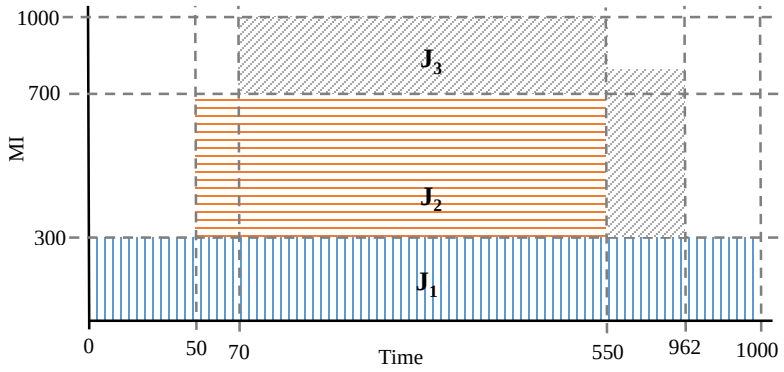

Figura 6. Distribución de la capacidad de procesamiento del contenedor para otro escenario posible.

Finalmente, se evaluaron las estrategias de asignación utilizando la degradación del desempeño y el perfil de desempeño (ver subsección 3.3) para cada estrategia a fin de eliminar los sesgos que podrían producirse al emplear promedios en la evaluación.

1.6 Estructura de la tesis

Esta sección detalla cómo se estructura el presente trabajo de tesis, describiendo brevemente los aspectos que se tratan en cada uno de los capítulos.

En el Capítulo 2, se presenta el estado del arte sobre el empleo de contenedores en la nube. Se detallan los fundamentos de esta investigación, se analizan trabajos relacionados y se presentan conceptos vinculados al tema tratado tales como: violaciones de SLA, el tiempo de terminación de todas las tareas (C_{max}) y modelos de energía.

El Capítulo 3 describe el planteamiento del problema, se especifica la notación matemática que sustenta la presente investigación y se definen los aspectos formales del problema planteado, así como las restricciones y consideraciones al respecto. Se presentan los métodos de evaluación utilizados y las estrategias de selección de los contenedores para la asignación de trabajos en los dos escenarios contemplados.

El Capítulo 4 detalla la configuración experimental utilizada y los resultados obtenidos. Se exponen los detalles del simulador utilizado, las cargas trabajo empleadas para las simulaciones y los resultados obtenidos en esta investigación.

En el Capítulo 5 se realiza la discusión de los resultados y las consideraciones fundamentales sobre estos en la presente investigación.

En el Capítulo 6 se presentan las conclusiones del trabajo de tesis, las contribuciones al conocimiento y las limitaciones de la tesis. Finalmente, se comenta el trabajo futuro asociado a la investigación.

Capítulo 2. Marco Teórico

En este capítulo se presentan los aspectos relacionados a la temática abordada en la presente tesis: el empleo de contenedores en los entornos de la nube, varias de sus características y particularidades. Se comentan los fundamentos de los acuerdos de nivel de servicio (SLA), un conjunto de trabajos que abordan el ahorro de energía en el cómputo en la nube, así como los modelos para su cálculo.

2.1 Contenedores

La computación en la nube puede ser definida como un modelo de acceso bajo demanda mediante una red de computadoras a un conjunto compartido de recursos computacionales configurables que incluye procesadores, redes, almacenamiento y memoria, que pueden ser rápidamente asignados y liberados con un mínimo esfuerzo de gestión o mínima asistencia por parte del proveedor de la nube (Mell and Grance 2011).

El uso de contenedores se ha hecho popular en los últimos años en los entornos de la nube, como se menciona en el capítulo anterior, con el modelo de Contenedores como Servicio. Dada la portabilidad y facilidad de implementación de los contenedores, es común su utilización en plataformas que ejecutan trabajos científicos. En (Celesti, Mulfari, Fazio, Villari, and Puliafito 2016) se utilizan contenedores específicamente enfocados en computadoras de placa única. Para ello, emplean dispositivos Raspberry Pi para ejecutar contenedores Docker. En ese trabajo se estudia la sobrecarga que produce la virtualización a nivel de sistema operativo en dispositivos de Internet de las Cosas (IoT). Como parte del estudio, se evidencia la pertinencia de usar contenedores para manipular sensores de IoT.

Varios estudios muestran el análisis de desempeño de nubes basadas en máquinas virtuales y en contenedores (Barik, Lenka, Rao, and Ghose 2017). Un contenedor provee una forma genérica de aislar un proceso del resto del sistema así como de todos los subprocesos hijos de éste (Dua, Raja, and Kakadia 2014). Algunas de las implementaciones de contenedores que se utilizan en la actualidad son: LinuX Container (LXC), que provee una interfaz de espacio de usuario sobre el kernel de Linux, y OpenVZ que utiliza una versión modificada del kernel de Linux con un conjunto de extensiones (Dua, Raja, and Kakadia 2014).

Introducidos por Google Cloud y Amazon Web Services, los contenedores pueden considerarse una nueva revolución en la era del cómputo en nube, ya que son ligeros, fáciles de configurar, gestionar y pueden reducir considerablemente su tiempo de arranque en comparación con las máquinas virtuales.

Un contenedor posee los mismos beneficios de asignación y aislamiento de recursos computacionales (CPU, por ejemplo) que una máquina virtual, pero tiene un enfoque diferente en cuanto a su estructura. Esto permite a los usuarios mayor facilidad para su portabilidad y eficiencia (Balasubramanian Sekar, Patil, Giusti, Bhide, and Gupta 2017).

La virtualización a nivel de sistema operativo, llamada también Contenedorización, utiliza dos tipos de contenedores, de sistema operativo y contenedores de aplicación. A continuación, se ofrece una breve descripción de ambos tipos. El diseño de algoritmos de asignación eficientes es el principal desafío para proveedores de nube basados en contenedores (Piraghaj 2016).

La Tabla 2 resume una taxonomía de la contenedorización referido a los componentes, interacción con el hardware y tecnologías disponibles entre ambos tipos de contenedores.

Tabla 2. Tipos de contenedores.

Virtualización	Componente	Comunicación con el hardware	Servicios (Tipo de tareas)	Tecnologías disponibles
A nivel de sistema operativo	Contenedor de sistema operativo (OS Container)	Llamadas estándar al sistema	De varios tipos	LXC, OpenVZ, Linux VServer, Jaulas FreeBSD y zonas Solaris.
	Contenedor de aplicación (App Container)		Un mismo tipo	Docket, Rocket

2.1.1 Contenedores de sistemas operativos

Este tipo de contenedores son entornos virtuales similares a las máquinas virtuales. Comparten el mismo kernel del ordenador donde se ejecutan, garantizando un entorno aislado de procesos que ejecutan. En ellos se pueden instalar, configurar y ejecutar diferentes aplicaciones, bibliotecas, etc., tal como lo haría con cualquier sistema operativo. De la misma forma que una máquina virtual, todo lo que se ejecuta dentro de un contenedor solo dispone de los recursos que se hayan asignado previamente a este entorno.

Este tipo de contendores pueden ejecutar múltiples procesos y servicios de manera similar a como lo hacen las máquinas virtuales, pero sin la necesidad de utilizar un hipervisor que emule el hardware ya que se realiza un aislamiento de los procesos y no de los recursos físicos. Estos elementos ofrecen un mejor desempeño y tiempos de inicio casi instantáneos.

El clúster de Google es un ejemplo de utilización de contenedores de sistemas operativos para ejecutar sus servicios. Esta empresa lanza más de 2 mil millones de contenedores por semana teniendo en cuenta todos sus centros de datos (Google 2014).

Este tipo de contenedores son usados en la plataforma Apache Mesos a fin de proporcionar el aislamiento requerido para la ejecución de cargas de trabajo. El kernel de Mesos se ejecuta en todos los servidores y proporciona aplicaciones con API para la administración de recursos y la programación en todo el centro de datos y entornos de nube. (Kumar, Andrews, Jayashankar, Mishra, and Suresh 2010) proponen una calendarización de dos niveles para esta plataforma llamada *recursos en oferta*. En el primer nivel, se encuentra un programador que se registra con el nodo maestro para que se ofrezcan recursos, y un proceso ejecutor que se inicia en los nodos esclavos para ejecutar las tareas.

2.1.2 Contenedores de aplicación

Los contenedores de aplicación están diseñados para ejecutar un solo servicio y generalmente se ejecutan sobre un contenedor de sistema operativo (Samo, Ahmed, and Shaikh 2017). La aparición de este tipo de estructura de virtualización ha revolucionado notablemente el cómputo en la nube. Docker, Rocket y Kubernetes son herramientas de gestión de contenedores utilizadas por casi la totalidad de los proveedores de servicios en la nube.

Este tipo de contenedores se utiliza preferentemente en los servicios tipo PaaS. La idea para el diseño de este tipo de contenedores es que se cree uno para cada componente de la aplicación. Este tipo de diseño funciona bien en los casos donde se desea crear aplicaciones distribuidas y sistemas multi-componentes basados en microservicios (Karle and Ci 2015).

2.2 Acuerdos de nivel de servicio (SLA)

Un acuerdo de nivel de servicio (SLA – Service Level Agreement) es un documento que establece las promesas de desempeño técnico hechas por un proveedor, y que incluye las compensaciones ante fallos en el desempeño (Badger, Patt-corner, and Voas 2012). Los acuerdos de nivel de servicio incluyen también los denominados objetivos de niveles de servicio (SLO – Service Level Objectives), que representan objetivos numéricos de calidad del servicio (Leitner, Ferner, Hummer, and Dustdar 2013).

La formulación de un SLA debe dejar claro aspectos tales como: la descripción del servicio que se brinda; la presentación de los niveles de servicio que se ofrecen; la manera en que se pueden monitorear los parámetros de servicio acordados, así como los formatos de los reportes; las penalizaciones en caso

de incumplimiento de los requerimientos. Las métricas asociadas a los parámetros del SLA se deben ajustar a los tipos de servicios y a los tipos de cargas de trabajo que utilicen estos servicios (Alhamad, Dillon, and Chang 2010a).

Los SLAs incluyen garantías respecto a la potencia computacional, el espacio de almacenamiento, el ancho de banda de las redes, la disponibilidad y la seguridad, entre otros. Los clientes deben pagar por el servicio recibido, pero los proveedores deben ser penalizados en caso de no cumplir con el servicio acordado. Por esta razón, una meta importante de un proveedor de servicios consiste en minimizar las violaciones de los acuerdos de nivel de servicio, para lo cual se deben adoptar estrategias apropiadas.

El análisis de las violaciones de los acuerdos de nivel de servicio debe partir, en primer lugar, de las características del tipo de servicio acordado, y tener en cuenta, además, qué tipo de compensaciones (o penalizaciones) se han establecido en el acuerdo.

En (Alhamad, Dillon, and Chang 2010a), definen un SLA simple que incluye cuatro parámetros de calidad del servicio:

- El plazo de terminación del trabajo.
- El tipo de restricción respecto al plazo de terminación del trabajo (el caso estricto, donde el usuario no admite retraso, y el caso flexible, en el cual se admite retraso, pero teniendo en cuenta la tasa de penalización establecida).
- La tasa de penalización.
- El presupuesto (la cantidad máxima que el usuario está dispuesto a pagar por la realización del trabajo).

Con estos elementos se propone la siguiente función de utilidad para cada trabajo i:

$$utilidad_i = presupuesto_i - (retraso_i * tasa\_penalizacion_i)$$

Para un trabajo i, existe retraso si el tiempo que toma para su finalización es mayor que el plazo acordado, es decir:

$$retraso_i = (tiempo\_finalizacion_i - tiempo\_inicio_i) - plazo_i$$

En caso de no existir retraso, el trabajo se completa antes, o en el plazo acordado, la utilidad coincide con el presupuesto. En caso contrario, la utilidad decrece linealmente y puede llegar a ser negativa, es decir, convertirse en una penalización (ver Figura 7).

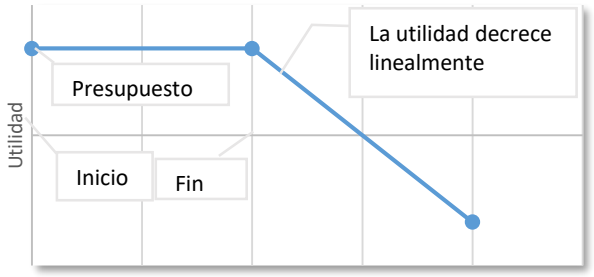

Figura 7. Modelo lineal de utilidad.

Un modelo lineal similar al anterior es utilizado en (Wu, Garg, and Buyya 2011) para el caso de la asignación de recursos en un tipo de servicio SaaS teniendo en cuenta como criterio la minimización de las violaciones de SLA. De igual manera, en (Garg, Gopalaiyengar, and Buyya 2011), se analiza el caso de asignación de recursos en presencia de tipos de cargas heterogéneas, con el objetivo de garantizar el cumplimiento de los SLA. También se utiliza un modelo lineal para caracterizar la penalización, tanto para cargas transaccionales, como para cargas no interactivas por lotes.

El monitoreo constante es una necesidad de los proveedores de servicio con el objetivo de ganar la confianza de los clientes. En (Ghuwairi, Salah, Alsarhan, Qudah, Qahmous, Baarah, and Oqaily 2018), se propone un modelo para el monitoreo de los SLA que pertenezcan a un mismo cliente y a un proveedor específico de múltiples máquinas virtuales, todos en un mismo ambiente.

Con la idea de disminuir las violaciones de los acuerdos de niveles de servicio, es posible también utilizar un enfoque predictivo, es decir, no solo aprender de los resultados pasados, si no, prevenir las violaciones en el futuro. En este sentido, en (Leitner, Ferner, Hummer, and Dustdar 2013), se propone un enfoque estadístico para la predicción de violaciones de SLA en tiempo de ejecución.

En (Sahal, Khafagy, and Omara 2016) se presenta una amplia revisión acerca de los diferentes enfoques propuestos para la gestión de las violaciones de SLA y se presentan aquellas áreas que requieren de investigaciones adicionales.

En este trabajo se contabiliza una violación de SLA cuando se le asigna a un trabajo una capacidad de procesamiento $s_j'(t)$ menor a la que demanda el trabajo, o sea s_j. La formulación matemática correspondiente se establece en la subsección 3.2.4.

2.3 Tiempo total de terminación

Si f_j denota el tiempo de terminación del trabajo j_j y J es el conjunto de n trabajos independientes, $\forall_j \in J$, es posible definir el tiempo total de terminación de todos los trabajos (*makespan*), denotado por C_{max}, de la siguiente manera:

$$C_{max} = max\{f_j\}, \forall_j \in J$$

El tiempo total de terminación de todos los trabajos se ha tomado como criterio en el proceso de calendarización en la nube en distintas propuestas.

Un algoritmo híbrido se propone en (Raju, Babukarthik, Chandramohan, Dhavachelvan, and Vengattaraman 2013), que toma como criterio la minimización del tiempo total de terminación de los trabajos.

En (Raju, Babukarthik, Chandramohan, Dhavachelvan, and Vengattaraman 2013) desarrollan un algoritmo de balance carga que permite utilizar de manera apropiada los recursos de la nube y reducir el tiempo total de terminación de las tareas.

En (Sathya Sofia and GaneshKumar 2018) se analiza un problema de la calendarización multiobjetivo que tiene en cuenta, tanto el consumo de energía, como el tiempo total de terminación de los trabajos. En la solución del problema de optimización multiobjetivo, se obtienen las soluciones no dominantes. Los resultados numéricos muestran que existen varias soluciones en el frente de Pareto óptimo de las cuales el usuario puede seleccionar las más apropiadas teniendo en cuentas diversos factores.

2.4 Modelos de energía

La potencia eléctrica es la proporción, o ritmo, por unidad de tiempo con la cual la energía eléctrica es transferida por un circuito y se expresa en vatios o watts (W). Los equipos eléctricos generan un consumo en función de la potencia que tengan y del tiempo que estén funcionando. El kilovatio hora (kWh) es la unidad de energía comúnmente utilizada, y equivale a la energía consumida por un equipo eléctrico cuya potencia fuese un kilovatio (kW) y estuviese funcionando durante una hora.

Es importante tener en cuenta la diferencia entre la potencia consumida y la energía consumida. Por ejemplo, el consumo de potencia puede ser disminuido al bajar el desempeño del CPU. Sin embargo,

en este caso, el programa tomará más tiempo en culminar y el consumo de energía eléctrica sería el mismo (Beloglazov 2013).

La eficiencia energética de un centro de datos se puede medir mediante la métrica de efectividad de uso de potencia (Power Usage Effectiveness - PUE), que expresa la razón entre el consumo total y el consumo del equipamiento asociado a las tecnologías de la información y las comunicaciones (equipos computacionales, equipamiento de redes, etc.). Este indicador toma valores mayores o iguales a la unidad. El valor promedio de la métrica PEU oscila alrededor de 1.8 en los centros de datos. Eso significa que la sobrecarga asociada a la infraestructura (climatización, distribución, etc.) representa 0.8 W por cada watt de potencia consumido en equipamiento computacional. Solo los grandes operadores reportan valores inferiores a 1.2 (Barroso, Clidaras, and Hölzle 2013).

Un estudio reciente realizado en Alemania, comentado en (Hintemann 2015), concluye que el consumo de energía de los centros de datos irá en ascenso si se mantienen las tendencias actuales (ver Figura 8).

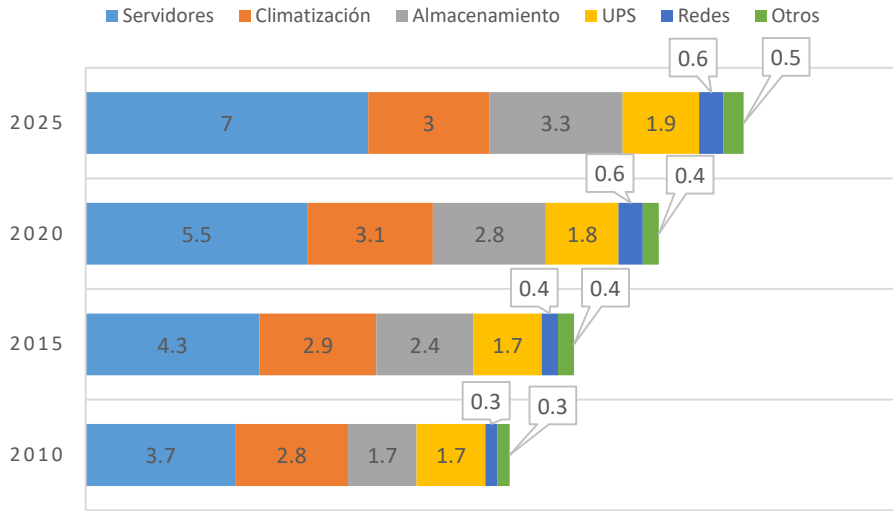

Figura 8. Consumo de energía eléctrica de los servidores y centros de datos de Alemania desde 2010 a 2015 y el pronóstico hasta 2025 (Hintemann 2015).

El consumo de energía puede ser reducido mediante dos técnicas: Gestión de la Potencia Estática (Static Power Management - SPM) y Gestión de Potencia Dinámica (Dynamic Power Management - DPM). El consumo de potencia estática tiene que ver con los componentes de bajo nivel presentes en los circuitos; mientras el consumo de potencia dinámica es creado por la actividad de los circuitos y depende, principalmente, del escenario de uso.

La técnica Escalamiento Dinámico de Voltaje y Frecuencia (Dynamic Voltage and Frequency Scaling - DVFS) es un ejemplo de gestión de potencia dinámica y propone una reducción combinada del suministro de voltaje y la frecuencia de reloj. El principal objetivo consiste en disminuir el desempeño del procesador cuando se encuentra ocioso. La mayoría de los procesadores modernos soportan esta técnica.

Dentro de un servidor, los principales consumidores de energía son: el procesador, la memoria interna y la memoria externa. Los continuos avances en el diseño de los procesadores, así como las técnicas de disminución de potencia (por ejemplo, DVFS) han llevado a que aumente su eficiencia con la posibilidad de consumir una fracción de su potencia total y mantener la capacidad de ejecutar programas. Por su parte, la gran cantidad de memoria presente en los servidores actuales, unido al hecho de no poseer los modos de trabajo con baja potencia, hace que ella sea uno de los componentes más importantes desde el punto de vista de su gestión eficiente respecto al consumo energético (Beloglazov 2013).

2.4.1 Modelos

Los modelos de energía tienen por objetivo predecir el consumo de potencia a partir de ciertas características o parámetros asociados al funcionamiento de un centro de datos.

En (Adhianto, Banerjee, Fagan, Krentel, Marin, Mellor-Crummey, and Tallent 2010), citando a (Fan, Weber, and Barroso 2007) y (Kusic, Kephart, Hanson, Kandasamy, and Jiang 2009), se plantea que el consumo de potencia de los nodos de un centro de datos puede ser descrito con bastante exactitud mediante una relación lineal entre el consumo de potencia y la utilización de los procesadores. Sin embargo, los servidores modernos están equipados con grandes cantidades de memoria, cuyo consumo de potencia comienza a ser dominante en el consumo total. Todo esto hace que la construcción de modelos analíticos precisos para los modernos procesadores multi-núcleos sea un problema de investigación complejo.

Los modelos lineales representan la manera más simple de estimar el consumo de potencia de un servidor. En (Pedram and Hwang 2010) y (Fan, Weber, and Barroso 2007), los autores suponen que el procesador es el único factor que influye en el consumo de potencia y proponen el siguiente modelo lineal aproximado:

$$P = P_{idle} + U * (P_{Peak} - P_{idle}) \quad (1)$$

donde, P representa el consumo total de potencia, P_{Peak} es el consumo pico, P_{idle} es el consumo cuando el servidor está desocupado (todos expresados en Watts), y U representa el porcentaje de utilización del procesador expresado con un valor entre 0 y 1.

En (Fan, Weber, and Barroso 2007) también se propone un modelo no lineal:

$$P = P_{idle} + (2 * U - U^r) * (P_{Peak} - P_{idle}) \quad (2)$$

En la ecuación anterior, r representa un parámetro de calibración que debe ser ajustado.

Los resultados muestran que el modelo lineal y el no lineal estiman con bastante exactitud el consumo de potencial real.

Inspirado en el modelo lineal anterior, en (Basmadjian, Ali, Niedermeier, De Meer, and Giuliani 2011) se propone uno que tiene en cuenta la cantidad de núcleos del procesador. Así, el consumo del i-ésimo núcleo es descrito de la siguiente manera:

$$P_{C_i} = P_{max} \frac{L_{C_i}}{100} \quad (3)$$

Aquí, P_{max} representa el consumo máximo (100% de utilización) de un núcleo, L_{C_i} indica el porcentaje de utilización del núcleo, y P_{C_i} el consumo de potencia de un núcleo.

A partir de la ecuación anterior, el consumo del procesador queda así:

$$P_{CPU} = P_{idle} + \sum_{i=1}^{n} P_{C_i} \quad (4)$$

donde n representa la cantidad de núcleos, P_{C_i} el consumo de un núcleo, y P_{idle} es el consumo cuando el servidor está desocupado.

Los autores destacan el hecho de que, en ciertos casos, el consumo no siempre es igual a la sumatoria de los consumos de los núcleos. Adicionalmente, se brindan varios modelos de consumo de potencia para otros componentes de un servidor (Basmadjian, Ali, Niedermeier, De Meer, and Giuliani 2011).

Un modelo similar, y equivalente, a la ecuación (4) es propuesto en (Liu, Zhan, and Zhang 2017), donde se incluye una tasa de consumo de potencia en estado inactivo del servidor.

$$P(u) = r_{idle} * \ P_{max} + (1 - r_{idle}) * P_{max} * u \quad (5)$$

En esta ecuación, r_{idle} representa la tasa de consumo de potencia en estado inactivo del servidor y u es la utilización del procesador del servidor $(0 \leq u \leq 1)$. Algunos estudios como (Adhianto, Banerjee, Fagan, Krentel, Marin, Mellor-Crummey, and Tallent 2010) y (Fan, Weber, and Barroso 2007) muestran que el valor de r_{idle} oscila alrededor de 0.7 de utilización.

En un ambiente real, la utilización del procesador (U) varía en el tiempo en función de la variabilidad de las cargas de trabajo, razón por la cual se representa como $U(t)$. De la misma manera, el consumo de potencia variará en el tiempo, es decir, sería $P\big(U(t)\big)$. Con esta notación, la energía total consumida por un servidor en un intervalo de tiempo $[t_0, t_1]$ sería igual a la integral del consumo de potencia en ese intervalo (Tian, He, Guo, Huang, Shi, Shang, Toosi, and Buyya 2018).

$$E = \int_{t_0}^{t_1} P\big(U(t)\big)dt \quad (6)$$

Se debe señalar que los modelos anteriores pueden ser usados para calcular el consumo de potencia (y el consumo de energía) con bastante exactitud si se dispone de los datos de utilización del procesador, así como los consumos en el estado de inactividad y de máxima utilización, en particular para cargas de trabajo que hacen uso intensivo del procesador. Algunos fabricantes publican esos valores de consumo (Beloglazov 2013).

Los modelos previos toman como base, únicamente, el nivel de utilización del procesador. Como se plantea en (Dhiman, Mihic, and Rosing 2010), esto puede ser insuficiente y proponen una metodología de modelado basada en mezclas gaussianas que incluye, además, otras características de las cargas de trabajo como, por ejemplo, la cantidad de instrucciones por ciclo (IPC – Instructions Per Cycle) y la cantidad de accesos a memoria por ciclo (MPC – Memory access Per Cycle). El modelo propuesto obtiene buenos resultados en los experimentos realizados (menor al 10% de error) y muestra la importancia de incluir otros parámetros como complementos a la utilización del procesador.

En el presente trabajo de tesis se utiliza el modelo de energía propuesto en (Armenta Cano, F. A., Chernykh 2018). Este modelo híbrido no lineal considera no solo la utilización del procesador, sino que emplea además la concentración de tareas como un elemento para la reducción del consumo de energía del CPU. En el modelo se reduce la energía al combinar tareas de diferentes tipos, tales como intensivas en el uso del CPU (CI) e intensivas en el uso de la memoria (MI). La base teórica que sustenta este modelo está disponible a profundidad en la sección 3.2.3. Como parte de este trabajo los autores muestran

mediante simulaciones y mediciones reales la relación entre la utilización del procesador y el consumo de energía de este componte. De igual forma, se contemplan las reducciones dadas por la combinación de tareas en un mismo CPU y el efecto que se logra mediante esta acción.

Todos los modelos revisados con anterioridad muestran una relación directa entre la utilización del CPU y la energía que éste consume. Dado el alto consumo energético de este componente de hardware.

Capítulo 3. Definición del problema

En este capítulo se presenta el planteamiento del problema, los modelos de infraestructura, trabajo y energía; se describen los objetivos de calendarización utilizados; la definición formal del problema y los criterios a considerar para su solución; se relatan los métodos de evaluación empleados para analizar la calidad de las soluciones obtenidas. Se describen las estrategias de asignación implementadas en los distintos niveles de acuerdo al estado en que se encuentra el sistema.

3.1 Planteamiento del problema

Los recursos computacionales de la nube deben aprovecharse al máximo. Para los proveedores de la nube, una utilización eficiente de su capacidad de cómputo se traduce en ahorros, por ejemplo, la reducción del consumo energético. Dado que dicha capacidad de cómputo es rentada por los clientes o usuarios, para el proveedor del servicio resulta fundamental mejorar continuamente la eficiencia en la ejecución de tareas y disponer de mayores recursos para continuar su contratación a nuevos clientes potenciales.

En los centros de datos, el problema de calendarización consiste en asignar un conjunto finito de recursos computacionales (memoria, CPU, red, etc.) a contenedores, para la ejecución de trabajos o tareas.

Como se muestra en (Lin 2013) para una correcta asignación de trabajos es necesario un algoritmo que administre el acceso a los distintos recursos computacionales (la capacidad de procesamiento - CPU). Existen diferentes tipos de tecnologías de manejo de éstos en el entorno de cómputo en la nube. Estas estrategias se implementan en varios niveles que disponen de parámetros como: costo, rendimiento, tiempo, prioridad, distancias físicas, ancho de banda, disponibilidad y utilización de recursos, entre otros. En el mismo artículo se señala que la eficiencia de la programación de tareas tiene un impacto directo en el rendimiento de todo el entorno de la nube; por tanto, es necesario optimizarla. En el entorno del cómputo en la nube, la programación se realiza en varios niveles tales como flujos de trabajo, máquinas virtuales, tareas, entre otros.

En (Kaneria 2016) se propone un algoritmo eficiente para la distribución de tareas en centros de datos. Se comprueba la disponibilidad de máquinas en éstos, así como de elementos de procesamiento y se asigna la tarea en consecuencia.

Otro elemento a considerar es el alto consumo energético que generan los centros de datos. En (Beloglazov and Buyya 2010) se presenta una arquitectura descentralizada del sistema de gestión de recursos que garantiza una gestión energéticamente eficiente de los centros de datos, que reduce los costos operacionales y que proporciona la calidad de servicio requerida al basarse en la migración de máquinas virtuales.

En (Lyu, Li, Yan, Masood, Sheng, and Luo 2017) señalan que los calendarizadores tienen como objetivo asignar tareas en los entornos de la computación en nube a una VM. En un esquema tradicional, el planificador de recursos asigna continuamente tareas a todas las máquinas virtuales. Estos recursos son: capacidad de procesamiento del procesador (CPU), memoria RAM, almacenamiento, ancho de banda. Los calendarizadores están sujetos a un conjunto de restricciones para realizar la asignación de recursos y mantener la calidad de servicio. Las restricciones mencionadas representan los requisitos que deben satisfacerse como: el mínimo poder de cómputo, mínimo ancho de banda de comunicación, máximo retardo y almacenamiento mínimo.

Los calendarizadores pueden dirigirse a objetivos diferentes: eficiencia energética y de refrigeración, balanceo de carga, calidad de servicio (QoS) o una combinación de éstos.

Tal y como se describe en (Giordano, Fiandrino, Kliazovich, Tchernykh, Giaccone, Guzek, and Bouvry 2016) otro elemento importante en este contexto es la asignación de tareas o trabajos a las nubes. Una asignación adecuada tiene un impacto significativo en el rendimiento de las aplicaciones y en los costos de operación de los centros de datos.

Otro elemento a destacar está asociado a la optimización y balanceo de carga en los contenedores, a fin de lograr un eficiente aprovechamiento de los recursos de cómputo e incrementar la cantidad de trabajos a realizar por éstos.

Como parte del presente trabajo, se desarrollan estrategias de asignación de tareas en contenedores de sistema operativo tomando como objetivos de optimización la minimización de energía, el tiempo de terminación del calendario y las violaciones de SLA en un centro de datos.

3.2 Definición formal

Esta sección presenta la formulación que sustenta la definición del problema tratado en la presente investigación.

3.2.1 Modelo de infraestructura

La infraestructura nube consiste de un conjunto de M servidores (procesadores) $P = \{p_1, p_2, .., p_M\}$. Cada servidor p_k, tiene una capacidad de procesamiento máxima Q_k. En estos servidores se ejecutan m contenedores $C = \{c_1, c_2, .., c_m\}$, la cantidad de contenedores por servidor se presenta en la subsección 4.2. Cada contendedor c_i, tiene una capacidad de procesamiento máxima q_i. La Tabla 3 presenta la notación relacionada a la infraestructura.

Tabla 3. Símbolos y definiciones del modelo de infraestructura.

Símbolo	Descripción
C	Conjunto de m contenedores $c_i, i = 1..m$
c_i	Contenedor asignado
$l_i\ (t)$	Capacidad de procesamiento asignada al tiempo t en el contenedor i
q_i	Capacidad de procesamiento máxima del contenedor c_i
P	Conjunto de M servidores $p_k, k = 1..M$
p_k	Servidor seleccionado
$J(c_i)$	Subconjunto de trabajos que se ejecutan en el contendor c_i
$C(p_k)$	Subconjunto de contenedores que funcionan en el servidor k
Q_k	Capacidad de procesamiento máxima de procesamiento del servidor k

3.2.2 Modelo de trabajos

Se consideran un conjunto de n trabajos independientes $J = \{j_1, j_2, .., j_n\}$. Cada trabajo $j_j = (r_j, s_j, w_j, \rho_j)$, tiene como características su tiempo de liberación(arribo al sistema) expresado en segundos $r_j \geq 0$, su capacidad de procesamiento requerida s_j expresada en MIPS, la cantidad de instrucciones de trabajo a ejecutar w_j y el tipo de trabajo ρ_j que indica si requiere para su ejecución un uso intenso de la CPU (CI) o de la memoria (MI). La Tabla 4 muestra la notación utilizada del modelo de trabajos.

Tabla 4. Símbolos y definiciones del modelo de trabajos.

Símbolo	Descripción
J	Conjunto de n trabajos $j_j, j = 1..n$
j_j	Trabajo a asignar
r_j	Tiempo de liberación del trabajo j
s_j	Capacidad de procesamiento requerida por el trabajo j (en MIPS)
$s_j'(t)$	Capacidad de procesamiento asignada al trabajo j en el instante de tiempo t
w_j	Cantidad de instrucciones del trabajo j (requerimientos computacionales necesarios para la ejecución del trabajo j, expresada en millones de instrucciones)
$w_j'(t)$	Cantidad de instrucciones pendientes por ejecutar del trabajo j al tiempo t
ρ_j	Tipo de trabajo (CI = intensivo en CPU, MI = intensivo en memoria)

3.2.3 Modelo de energía

Otro criterio para la asignación de tareas radica en minimizar el consumo de energía del centro de datos de la nube. En (Piraghaj, Dastjerdi, Calheiros, and Buyya 2015) se modela el entorno de Contenedores como Servicio (CaaS), se aborda el problema de optimización de energía asociada a disminuir el número de servidores en funcionamiento y se define el modelo de energía de un centro de datos. La Tabla 5 presenta la muestra la notación utilizada para la formulación de la energía.

Tabla 5. Símbolos y definiciones del modelo de energía.

Símbolo	Descripción
E	Consumo de energía total del centro de datos
$E^{op}(t)$	Consumo de energía del centro de datos al tiempo t
$e_k^{proc}(t)$	Consumo de potencia del procesador k al tiempo t
e_k^{idle}	Potencia inactiva o potencia base del procesador k
e_k^{used}	Potencia de uso del procesador k
e_k^{max}	Consumo máximo de potencia del servidor k
M	Cantidad de servidores del centro de datos
m	Número de contenedores
$U_T(t)$	Utilización del procesador(CPU) del servidor k al tiempo t
$U_{CI}(t)$	Utilización de tareas que hacen un uso intenso del CPU al tiempo t
$U_{MI}(t)$	Utilización de tareas que hacen un uso intenso de la memoria al tiempo t
$\varphi_{CI}(t)$	Concentración de tareas con uso intensivo del CPU en el servidor k al tiempo t
$\varphi_{MI}(t)$	Concentración de tareas con uso intensivo la memoria en el servidor k al tiempo t

El *makespan* o C_{max} define el tiempo de terminación del calendario o tiempo de finalización de la última tarea que se ejecuta en el conjunto de m contenedores.

El consumo de energía del centro de datos se define como la integral de la energía consumida en el tiempo de operación C_{max}:

$$E = \int_{t=1}^{C_{max}} E^{op}(t)dt \tag{7}$$

El consumo de energía del centro de datos en el tiempo t se calcula de la siguiente manera:

$$E^{op}(t) = \sum_{k=1}^{M} e_k^{proc}(t) \tag{8}$$

El consumo de potencia del procesador k al tiempo t se calcula como la suma de e_{idle} (el procesador está encendido, pero no se usa) y $e_{used}(t)$ (el procesador está encendido y ejecuta trabajos:

$$e_k^{proc}(t) = o(t)\left(e_k^{idle} + e_k^{used}(t)\right) \tag{9}$$

Donde el valor de $o(t) = 1$ si el procesador k está funcionando al tiempo t y $o(t) = 0$ en otro caso.

Y el consumo de potencia del servidor i al tiempo t se calcula como la diferencia entre la potencia máxima (e^{max}) y la potencia base (e^{idle}). Esta diferencia se ajusta con dos valores, el primero que representa la suma de la porción de potencia consumida por cada tipo de trabajo $F(t)$, y el segundo representa la fracción de consumo de potencia cuando dos o más aplicaciones se ejecutan en el mismo procesador $g(\varphi_{CI}(t))$:

$$e_k^{used}(\mathrm{t}) = \left(e_k^{max} - e_k^{idle}\right) * F(t) * g(\varphi_{CI}(t)) \tag{10}$$

Para el alcance de esta investigación se consideran dos tipos de trabajos: los que demandan un uso intenso de la CPU (CI) y los que requieren un uso intenso de la memoria (MI). Tal y como se describe en (Armenta Cano, F. A., Chernykh 2018), el cálculo de $F(t)$ se obtiene a partir de la fracción de consumo de potencia cuando el procesador ejecuta cada tipo de trabajo $f_d(U_d(t))$:

$$F(t) = \sum_{\forall d} f_d(U_d(t)), 0 \le F(t) \le 1, d \in \{CI, MI\} \tag{11}$$

Para la estimación del consumo de energía de cada servidor k, se considera la utilización del CPU, porque éste es el componente que presenta la mayor correlación relativa al consumo de energía con respecto a su tasa de utilización. La utilización total de cada host al tiempo t se define por:

$$U_T(\mathrm{t}) = U_{CI}(t) + U_{MI}(t) \tag{12}$$

Donde:

$$U_{CI}(t) = \frac{\sum_{\forall i \in C(p_k)} \sum_{\forall j \in J(c_i)} s_j'(t)}{Q_k} \; tal\ que \left\{ \forall_j \in J(c_i) \mid \rho_j = CI \right\} \tag{13}$$

$$U_{MI}(t) = \frac{\sum_{\forall i \in C(p_k)} \sum_{\forall j \in J(c_i)} s_j'(t)}{Q_k} \; tal\ que \left\{ \forall_j \in J(c_i) \mid \rho_j = MI \right\} \tag{14}$$

El ajuste del valor de la energía utilizada por el procesador tiene como último componente la función de concentración de tareas $g(\varphi_{CI}(t))$. La función $\varphi_{CI}(t)$ define la concentración de los trabajos intensivos en CPU que se están ejecutando en el procesador al tiempo t. Dado que en el presente trabajo solo se consideran dos tipos de tarea, se puede afirmar:

$$\varphi_{CI}(t) = \frac{U_{CI}(t)}{U_T(t)} \tag{15}$$

$$\varphi_{MI}(t) = 1 - \varphi_{CI}(t) \tag{16}$$

Esta función representa la fracción de potencia consumida por el procesador cuando se combinan trabajos de tipos CI y MI y la utilización total del procesador k está al 100% en el instante de tiempo t, es decir, $U_k(t) = 1$.

En (Armenta Cano, F. A., Chernykh 2018) refieren que si el procesador ejecuta diferentes tipos de trabajos con la misma utilización de CPU, cada tipo de trabajo contribuye de manera diferente al consumo total de energía. Si no se considera la combinación entre aplicaciones, la contribución de potencia de cada aplicación en el procesador se define por separado, por lo tanto, $g\big(\varphi_{CI}(t)\big) = 1$.

3.2.4 Criterios de optimización

El problema que se aborda en el presente trabajo consiste en asignar los n trabajos del conjunto de J a un conjunto C de m contenedores para optimizar los objetivos: C_{max}, consumo de energía y las violaciones de SLA.

En el instante de tiempo $t = r_j$, el trabajo j_j está disponible para su ejecución. En ese momento, se asigna a un contenedor c_i. Cada contenedor está identificado por su capacidad de procesamiento q_i. Como la capacidad de procesamiento asignada a cada trabajo $s'_j(t)$ puede variar en el tiempo. Se define la capacidad de procesamiento asignada al trabajo j como:

$$s'_j(t) = q_i - l_i(t) \tag{17}$$

La ecuación (17) define que la capacidad de procesamiento asignada al trabajo j al tiempo t es igual a la capacidad de procesamiento del contenedor seleccionado menos la capacidad de procesamiento previamente asignada. Esta capacidad de procesamiento se define como la suma de las capacidades de procesamientos asignadas a los trabajos que se ejecutan en el contenedor al tiempo t y se define por la siguiente expresión:

$$l_i(t) = \sum_{\forall j \in J(c_i)} s'_j(t) \tag{18}$$

En el caso de estudio que se presenta en esta tesis, la calendarización (scheduling) puede pensarse como la asignación de los n trabajos a la infraestructura de nube (ver 3.2.1). Una violación de SLA en el contenedor i, ocurre cuando el trabajo j demanda una capacidad de procesamiento s_j y se le asigna una capacidad de procesamiento menor $\big(s'_j(t)\big)$, al no cumplirse este requerimiento se contabiliza una violación de la siguiente manera:

$$\propto_{i,j}(t) = \begin{cases} 1 & Si\ s'_j(t) < s_j \\ 0 & en\ otro\ caso \end{cases} \quad (19)$$

Para obtener el número total de violaciones del centro de datos se contabilizan todas las ocurridas para cada uno de los m contenedores:

$$SLA = \sum_{i=1}^{m} \sum_{j \in J(c_i)} \propto_{i,j}(t), \forall_i \in C, \forall_j\ trabajos\ asignados\ al\ contenedor\ c_i \quad (20)$$

Con el fin de minimizar el consumo de energía del centro de datos, las violaciones de SLA y el tiempo de terminación de todas las tareas C_{max}, se plantea como meta del presente trabajo:

$$\text{minimizar}\ \{C_{max}, E, SLA\} \quad (21)$$

3.3 Métodos de evaluación

Esta sección hace referencia a los métodos empleados para evaluar la calidad de una solución obtenida por una estrategia. Además, se definen las particularidades de estos métodos y los aspectos que permiten comparar la calidad de una solución con otra.

La mayoría de los problemas implican la optimización simultánea de más de un objetivo, a menudo éstos compiten entre sí o se encuentran en conflicto. Una de las diferencias entre la optimización de un solo objetivo, es que en ésta la solución suele estar claramente definida, por ejemplo: maximizar las ganancias de una empresa o minimizar las pérdidas de un grupo de trabajo. Sin embargo, esto no suele ser válido para problemas donde se desea optimizar más de un objetivo.

Los problemas de optimización con múltiples objetivos son comunes, por ejemplo: como adquirir un equipo con una buena calidad y bajo costo, o el que se presenta en este trabajo, encontrar estrategias que minimicen las violaciones SLA, el C_{max} y el consumo de energía de un centro de datos.

3.3.1 Degradación del desempeño

En (Tchernykh, A., Schwiegelsohn, U., Yahyapour, R., Kuzjurin 2010) introducen una solución que supone la misma importancia para cada métrica, lo cual es adecuado para el análisis de esta investigación, ya que se considera la misma importancia para la minimización de las violaciones de SLA, el tiempo de terminación de todas las tareas y el consumo energético.

El objetivo es encontrar una estrategia que tenga un buen desempeño en todos los casos de estudio, con la expectativa de obtener un buen desempeño en otras condiciones, por ejemplo, con diferentes configuraciones de infraestructura y cargas de trabajo.

Para obtener una guía efectiva al elegir la mejor estrategia, se realiza un análisis del consumo de energía, el tiempo de terminación del calendario y las violaciones de SLA de acuerdo con la metodología de degradación promedio propuesta en (Tsafrir, Etsion, and Feitelson 2007). Dicha metodología se ha aplicado a problemas de calendarización en trabajos como: (Armenta Cano, F. A., Chernykh 2018) , (Cortés Mendoza and Chernykh 2018) y (Lozano Guzmán, L. M., Chernykh 2013).

Como primer paso, se evalúa la degradación del desempeño (error relativo) de cada estrategia bajo alguna métrica. Esta evaluación se realiza tomando como base la estrategia con el mejor desempeño para la métrica.

$$(\gamma - 1) * 100 \text{ donde } \gamma = \frac{valor\ de\ la\ métrica}{mejor\ valor\ obtenido\ en\ la\ métrica} \tag{23}$$

para el $valor\ de\ la\ métrica > 0$

Con posterioridad, se promedian y clasifican los valores de degradación de desempeño de las estrategias. La mejor estrategia es aquella con la degradación promedio más baja con respecto al rendimiento; esta tiene el primer puesto.

Se presentan los promedios de degradación de los valores de las métricas estudiadas para evaluar el desempeño de las estrategias implementadas y mostrar si alguna domina al resto. Para eliminar la influencia de una pequeña porción de datos con una gran desviación en el proceso de marcaje y ayudar con la interpretación de los datos, se puede utilizar el perfil de desempeño.

3.3.2 Perfil de desempeño

Al sustentar conclusiones basadas en promedios pueden existir casos donde dominen pequeñas instancias del problema con alta desviación. Para analizar los efectos negativos de trabajar con promedios se presentan los perfiles de desempeño de las estrategias implementadas.

El perfil de desempeño $\delta(\tau)$ es una función constante, no decreciente, que define la probabilidad de que la razón de desempeño $\gamma = \frac{valor\ de\ la\ métrica}{mejor\ valor\ obtenido\ en\ la\ métrica}$ esté dentro de un factor τ alejado de la mejor solución (Dolan and More 2001). Con el $valor\ de\ la\ métrica > 0$.

La ecuación (24) muestra cómo se calcula el perfil de desempeño de cada métrica, donde $resultado$ es la degradación del desempeño de cada solución encontrada y τ es el factor de degradación que se evalúa en cada caso, para saber qué porcentaje de las soluciones están por debajo de dicho factor. Una vez que se dispone de la cantidad de resultados que están por debajo del factor τ evaluado, se divide entre el número total de experimentos $n_{experimentos}$

$$|\delta_{metrica}| = \frac{|\{i|resultado_i \leq \tau\}|}{n_{experimentos}} \tag{24}$$

La función $\delta(\tau)$ es la función de distribución acumulativa. Estrategias con alta probabilidad de $\delta(\tau)$ para valores pequeños de τ son preferibles.

3.3.3 Modelo de las sumas ponderadas

Este método consiste en construir un problema de un solo objetivo asociando un peso a cada objetivo. En la ecuación (25) se asigna a cada métrica un peso relativo. De esta forma se puede reducir la complejidad del análisis ya que se pasa de un problema de optimización multiobjetivo a uno de minimización monobjetivo. Al problema de minimización que considera las 3 métricas, se le asocia un peso ponderado a cada métrica y se busca minimizar la función F. De esta forma el problema ponderado es:

$$F_O = w_1 * C_{max} + w_2 * SLA + w_3 * E \tag{25}$$

$$\sum_{i=1}^{3} w_i = 1 \qquad 0 \leq w_i \leq 1 \tag{26}$$

$$minimizar(F_O) \tag{27}$$

Donde $w_1, w_2\ y\ w_3$ definen los pesos respectivos de cada métrica. La asignación del valor de las ponderaciones es la clave de esta técnica. En muchos problemas la asignación de las ponderaciones puede realizarse de manera que todos los pesos sean iguales, esto significa que todos los valores de las métricas tengan el mismo peso. Otro mecanismo de asignación de pesos suele ser dejar a un decisor externo la asignación de los pesos en función de sus consideraciones. En este trabajo no se emplea este método de evaluación.

3.4 Estrategias de selección del contenedor

Existen dos escenarios posibles relativos a la capacidad de procesamiento disponible en el conjunto m de contenedores. El primer escenario coincide con lo que adelante se nombra primer nivel de asignación y se refiere a cuando al menos un contenedor posee capacidad de procesamiento suficiente para satisfacer la requerida por el trabajo que se asigna.

3.4.1 Modelo con capacidad requerida del contenedor

La Figura 9 muestra una representación de dos contenedores c_0 (izquierda) y c_1 (derecha). En particular se ilustra la asignación del trabajo j_3 al contenedor c_0 identificado por la capacidad de procesamiento q_0. Nótese que ambos contenedores poseen la capacidad de procesamiento suficiente en el instante t_k para cubrir lo que demanda este trabajo, esto se aprecia dado que el largo del trabajo (asociado a la capacidad de procesamiento requerida) no varía. Por tal motivo, el trabajo se asigna al contenedor 0 y no se generan violaciones de SLA.

Esta situación puede verse como una variante del problema de empaquetamiento de contenedores (Bin Packing); es posible comparar los trabajos con los elementos (ítems) y los contenedores en la nube como los contenedores (bins). En (Pandit, Chattopadhyay, Chattopadhyay, and Chaki 2014), se realiza una comparación similar entre la asignación de cargas de trabajo a máquinas virtuales y el problema de empaquetamiento.

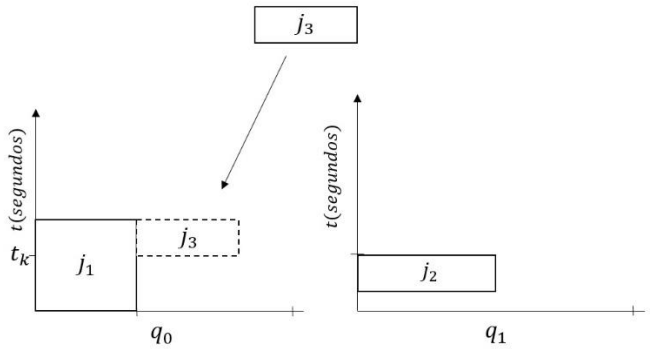

Figura 9. Asignación del trabajo a contenedor con capacidad de procesamiento suficiente disponible para j_3.

Otro escenario ocurre cuando ninguno de los contenedores tiene disponible la capacidad de procesamiento requerida por el trabajo en el momento de la asignación. La Figura 10 muestra esta situación, al asignarse el trabajo j_3 a los contenedores activos c_0 (izquierda) y c_1 (derecha). Ningún contenedor puede satisfacer lo demandado para este trabajo, esto se aprecia dado que el largo del trabajo (asociado a la capacidad de procesamiento requerida), la política de asignación (de segundo nivel) asigna el trabajo al contenedor 0. En el instante t_k, se les asigna una capacidad de procesamiento $s_j'(t)$ menor a

la demandada por el trabajo j_3. Al presentarse esta situación, se producen dos violaciones de SLA, una por cada trabajo que se ejecuta en c_0.

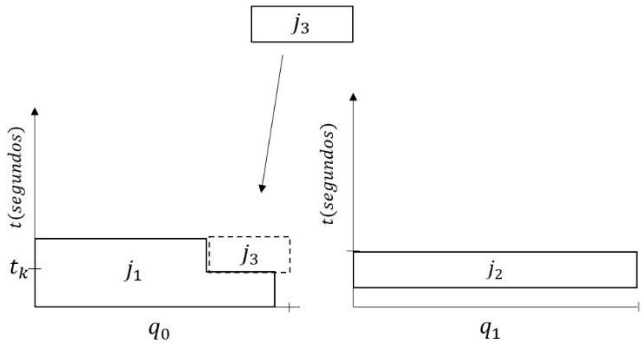

Figura 10. Asignación del trabajo a contenedor sin capacidad de procesamiento suficiente disponible para j_3.

La asignación de j_3 en cualquiera de los contenedores desencadenara el procedimiento de reasignación de CPU, este distribuye equitativamente la capacidad de procesamiento entre los trabajos del contenedor.

A continuación, se describen las estrategias implementadas para la asignación del trabajo en los contenedores. En tal sentido, se definen tres niveles de asignación que seleccionan a qué contenedor será enviado el trabajo para su ejecución.

3.4.1.1 Primer nivel de asignación

Este nivel está relacionado a la asignación del trabajo cuando hay al menos un contenedor con capacidad de procesamiento para satisfacer la capacidad de procesamiento mínima requerida por el trabajo j_j. La Tabla 6 describe las estrategias que se utilizan si existe más de un contenedor que cumple con el requerimiento anterior. En este nivel no existen violaciones de SLA dado que se puede satisfacer la demanda mínima del trabajo.

Tabla 6. Estrategias de asignación del primer nivel.

Estrategia	Descripción
Aleatorio (Random)	Asigna aleatoriamente el trabajo j a un contenedor utilizando una distribución uniforme en el rango $[1..k] \mid k \leq m$; k es el número de trabajos que se encuentran en ejecución en el contenedor previo al arribo del trabajo j.
Round Robin (RR)	Asigna el trabajo j al contenedor disponible y capaz de ejecutarlo mediante la estrategia Round Robin.
Primer ajuste (First Fit)	Asigna el trabajo j al primer contenedor disponible y capaz de ejecutarlo.
Mejor ajuste (Best Fit)	Asigna el trabajo j al contenedor que minimiza la diferencia entre la capacidad de procesamiento q_i y la suma de las capacidades de CPU asignadas a las tareas que se

	están procesando, más la capacidad de procesamiento requerida por el trabajo j_j. $Min\left(q_i - \left(\sum s_d + s_j\right)\right) \geq 0$ para todos los d trabajos asignados al contenedor c_i.
Peor ajuste (Worst Fit)	Asigna el trabajo j al contenedor que maximiza la diferencia entre la capacidad de procesamiento q_i y la suma de las capacidades de CPU asignadas a las tareas que se están procesando, más la capacidad de procesamiento requerida por el trabajo j_j. $Max\left(q_i - \left(\sum s_d + s_j\right)\right) \geq 0$ para todos los d trabajos asignados al contenedor c_i.
Máxima cantidad de violaciones SLA (MaxSla)	Asigna el trabajo j al contenedor donde han ocurrido más violaciones SLA.
Contenedor con menor cantidad de trabajo pendiente (MinAW)	Asigna el trabajo j al contenedor con menor cantidad de instrucciones pendientes por ejecutar en el instante de tiempo $t = r_j$. $Min\left(\sum w_j'\left(r_j\right)\right)$ $\forall_{j \in J}$

3.4.1.2 Segundo nivel de asignación

Las estrategias que se utilizan en este nivel se ejecutan si existe al menos un contenedor que no tenga ocupada toda su capacidad de procesamiento q_i. En este nivel ocurren siempre violaciones de SLA. Las estrategias en este nivel se describen en la Tabla 7.

Tabla 7. Estrategias del segundo nivel.

Estrategia	Descripción
Mínimo número de tareas en ejecución (MinTask)	Asigna el trabajo j al contenedor con menor número de trabajos en ejecución entre todos aquellos que tienen capacidad de procesamiento disponible.
Máxima Capacidad de procesamiento disponible (MaxCap)	Asigna el trabajo j al contenedor con mayor capacidad de procesamiento disponible.
Mínima Capacidad de procesamiento disponible (MinCap)	Asigna el trabajo j al contenedor con menor capacidad de procesamiento disponible.
Mínima cantidad de violaciones SLA (MinSla)	Asigna el trabajo j al contenedor donde han ocurrido menos violaciones SLA.

3.4.1.3 Tercer nivel de asignación

El tercer nivel de asignación se refiere al escenario en el cual los m contenedores están utilizando su máxima capacidad de procesamiento q_i. En este nivel, el trabajo asignado generará un incremento de las violaciones de SLA en el contenedor donde se ubique. La redistribución de la capacidad de procesamiento se realiza de manera equitativa entre todos los trabajos que se encuentran en ejecución en el contenedor donde se asignó el trabajo. Este tipo de distribución proporcional se realiza a fin de no incrementar significativamente el tiempo de terminación de las tareas. Las estrategias empleadas a este nivel son: MinSla y MinTask definidas en el segundo nivel.

El Algoritmo 1 describe el funcionamiento de la estrategia Best Fit – MaxCap – MinTask. Inicialmente la estrategia de primer nivel trata de asignar el trabajo a los contenedores activos sin incrementar las violaciones de SLA mediante el mejor ajuste (línea 2), por lo que los contenedores deben tener espacio suficiente para procesar la tarea.

Si no es posible asignar el trabajo sin incrementar las violaciones de SLA, se inicializa un contenedor más con la condición que haya suficientes recursos físicos para la ejecución del contenedor (líneas 4 - 7).

Algoritmo 1. Best Fit – MaxCap - MinTask

Entrada: Lista de contenedores activos ACT y trabajo $\boldsymbol{i}$ ($\boldsymbol{j_i}$)
Salida: Identificador del contenedor donde se ejecuta $\boldsymbol{j_i}$

```
1   ctIndex ← -1
2   ctIndex ← Best_Fit (ACT, j_i)
3     si ctIndex < 0 entonces
4       si se puede crear un contenedor nuevo entonces
5         crear nuevo contenedor nCT
6         ctIndex ← identificador de nCT
7         agregar nCT a la lista ACT
8       si no
9         ctIndex ← MaxCap ()
10        si ctIndex < 0 entonces
11          ctIndex ←MinTask()
12        reasignar procesamiento en contenedor ctIndex
13  regresar ctIndex
```

Las estrategias de segundo nivel determinan el destino del trabajo (línea 9) cuando no es posible inicializar nuevos contenedores y ningún contenedor tiene recursos suficientes para procesar el trabajo. En esta situación es inevitable afectar el SLA de los trabajos en ejecución. En este ejemplo, la función MaxCap() busca el contenedor con mayor capacidad disponible para asignar el trabajo.

Finalmente, las estrategias de tercer nivel determinan el destino del trabajo (línea 11) cuando todos los contenedores están llenos. Similar al caso anterior, es inevitable afectar el SLA de los trabajos en ejecución. En este ejemplo, la función MinTask() busca el contenedor con menor número de trabajos asignados con el objetivo de minimizar el número de tareas afectadas.

Cuando se asigna un trabajo en un contenedor sin espacio suficiente entonces se debe distribuir la capacidad de procesamiento del contenedor proporcionalmente entre las tareas (línea 12).

3.4.2 Modelo con capacidad completa del contenedor

La Figura 11 muestra la asignación del trabajo j_1 al contenedor c_0. Para este escenario se le asigna toda la capacidad de procesamiento q_0 al trabajo mencionado, por lo tanto, el contenedor siempre está lleno cuando tiene asignado por lo menos un trabajo. Nótese que en este caso en particular, el trabajo j_1 recibe más capacidad de procesamiento que la que demanda para su ejecución; esta asignación por encima de lo requerido permite que el trabajo culmine antes de lo esperado.

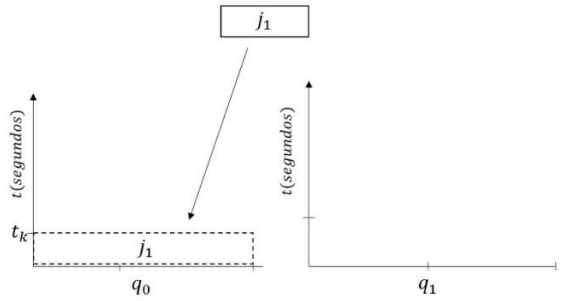

Figura 11. Asignación del trabajo a contenedor sin trabajos en ejecución (escenario 2).

Para este escenario se definen dos niveles relacionados con el proceso de asignación. Las estrategias de primer nivel permiten asignar los trabajos que llegan al sistema dentro de los contenedores encendidos. La condición que se debe cumplir es que la asignación del trabajo dentro del contenedor no genere violaciones de SLA. El calendarizador considera la velocidad mínima de todos los trabajos en el contenedor para determinar si se puede asignar más trabajos en el contenedor. Esto se traduce en una reducción de velocidad de procesamiento de los trabajos en ejecución sin generar ninguna violación de SLA.

La Figura 12 muestra la asignación del trabajo j_2 al contenedor c_0 (izquierda) representado por su capacidad de procesamiento q_0. En el momento t_k, c_0 está ejecutando j_1 con una velocidad de $s'_{j_1}(t_k)$ (recuadro punteado) superior a la velocidad mínima requerida s_{j_1}(recuadro rojo). A la llegada de j_2, el calendarizador determina que $s_{j_1} + s_{j_2} \leq q_0$ por lo tanto se puede asignar j_2 al contenedor c_0 sin generar violaciones de SLA en los trabajos asignados a este. Después de la asignación de j_2 en c_0, la velocidad de j_1 es ajustada ($s'_{j_1}(t_{k+1})$) pero sin que esta sea menor a la velocidad mínima.

Las estrategias de segundo nivel deciden el destino de los trabajos que al ser asignados generan violaciones de SLA, esta situación se produce cuando todos los contenedores están procesando trabajos y ninguno puede proporcionar la capacidad mínima de procesamiento solicitada por el trabajo que llega al sistema.

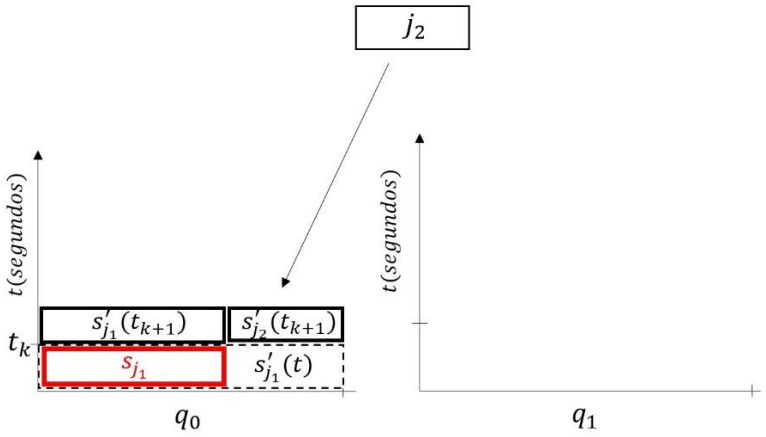

Figura 12. Asignación del trabajo a contenedor (escenario 2).

Para este modelo, la Tabla 5 presenta las estrategias de primer nivel empleadas en la experimentación. MinSla y MinTask definen las estrategias de segundo nivel.

El Algoritmo 2 muestra el funcionamiento de la estrategia Best Fit – MinTask. Inicialmente la estrategia de primer nivel trata de asignar el trabajo a los contenedores activos sin incrementar las violaciones de SLA mediante el mejor ajuste (línea 2).

En este escenario, los contenedores activos siempre están llenos porque los trabajos reciben una mayor capacidad de cómputo a la solicitada. Por lo tanto, la estrategia de mejor ajuste utiliza la capacidad de procesamiento mínima requerida por los trabajos para determinar si es posible asignar un trabajo más.

Algoritmo 2. Best Fit – MinTask	
Entrada: Lista de contenedores activos ACT y trabajo $\boldsymbol{i}$ ($\boldsymbol{j_i}$)	
Salida: Identificador del contenedor donde se ejecuta $\boldsymbol{j_i}$	
1	ctIndex ← -1
2	ctIndex ← **Best_Fit** (ACT, $\boldsymbol{j_i}$)
3	**si** ctIndex < 0 **entonces**
4	**si** se puede crear un contenedor nuevo **entonces**
5	crear nuevo contenedor nCT
6	ctIndex ← identificador de nCT
7	agregar nCT a la lista ACT
8	**si no**
9	ctIndex ←**MinTask**()
10	**reasignar procesamiento** en contenedor ctIndex
11	regresar ctIndex

Capítulo 4. Configuración experimental y resultados

En este capítulo se presenta la configuración experimental, el marco de simulación utilizado para realizar los experimentos y las cargas de trabajo empleadas. Se describe además la metodología utilizada para el análisis. Todos los experimentos se realizan en el simulador CloudSim (Calheiros, Ranjan, Beloglazov, De Rose, and Buyya 2011). La arquitectura del centro de datos se define con una topología de dos niveles con M servidores en el primer nivel y m contenedores en el segundo nivel.

Como parte de este capítulo se presentan los resultados obtenidos. Para el primer escenario, se implementaron 58 variantes de estrategias, obtenidas a partir de las combinaciones de las 12 estrategias básicas y los 3 niveles. Para el segundo escenario se implementaron 14 variantes de estrategias, obtenidas a partir de las combinaciones de 7 estrategias y 2 niveles de asignación. La implementación de las estrategias se realiza en el simulador CloudSim con cargas de trabajo reales.

4.1 Marco de simulación

Con el objetivo de evaluar las estrategias implementadas y simular un ambiente nube real, es necesario utilizar un entorno controlado que permita realizar experimentos, visualizar y evaluar los resultados. Se efectuó un análisis de varios simuladores nube en el estado del arte. Estos permiten modelar y simular aspectos fundamentales de los entornos de la nube como: elementos de infraestructura, trabajos, calendarizadores, entre otros. Los entornos de simulación pueden acelerar el proceso de desarrollo de una investigación, como la presente, ya que permiten la realización de experimentos repetibles en entornos controlados (Wang and Wu 2009). A continuación, se presentan algunos de los simuladores existentes.

MDCSim (Lim, Sharma, Nam, Kim, and Das 2009) es un simulador empleado para el modelado de entidades en centros de datos de varios niveles. La estimación del consumo de energía del equipamiento simulado es un aspecto muy importante implementado en este simulador. MDCSim contempla la simulación del tráfico de red modelado como un grafo dirigido y considera métricas como el rendimiento y el tiempo de respuesta. Se pueden modelar componentes prototipo a nivel de servidor, a nivel de aplicaciones y a nivel de servidor de base de datos.

GDCSim (Chen, Vasardani, and Winter 2017) se implementa basándose en la concepción de un centro de datos que respeta el cuidado del medio ambiente. Este simulador se diseñó para permitir a los proveedores de servicios de la nube medir el impacto energético de diferentes tipos de diseños físicos de

infraestructura y algoritmos de administración de recursos, antes de hacer un despliegue real. Los resultados muestran cómo los recursos computacionales hacen uso eficiente de la energía y las simulaciones utilizan estrategias para disminuir el consumo energético; tales como algoritmos de asignación que favorecen el uso racional de los recursos. GDCSim puede extenderse con nuevos modelos de consumo de energía, recursos de hardware, administración de recursos, entre otros.

GroudSim (Ostermann, Plankensteiner, Prodan, and Fahringer 2011) es otro simulador que modela el entorno de cómputo en la nube. Esta herramienta, implementada en Java y basada en eventos, proporciona información estadística sobre la simulación una vez terminada la misma. GroudSim brinda la posibilidad de registrar fallas dentro de ciertos intervalos de tiempo tal y como puede ocurrir en escenarios reales.

CloudSim (Calheiros, Ranjan, Beloglazov, De Rose, and Buyya 2011) es uno de los simuladores más utilizados en la realización de estudios asociados a la computación en la nube. Desarrollado en Java por el laboratorio de cómputo en la nube y sistemas distribuidos (CLOUDS, por sus siglas en inglés), de la Universidad de Melbourne, Australia. Destaca por su facilidad de uso y la amplia documentación que posee. Este simulador permite modelar nubes, agentes de servicios, políticas de aprovisionamiento y asignación de recursos. En esta investigación, se utiliza la versión 3.3 de este simulador con la implementación de clases que permiten la obtención de reportes asociados a: calendarios de asignación, ejecución de tareas, estadísticas de la simulación y valores de energía asociados a este proceso. La arquitectura de CloudSim brinda la posibilidad de modificar, a nivel de usuario, los requerimientos necesarios para la simulación y las políticas de asignación.

4.2 Configuración de la simulación

El ambiente de simulación es muy importante porque permite validar la eficiencia de las estrategias en ambientes reales. En esta tesis se analiza el desempeño de las estrategias con una configuración de nube con recursos limitados y representativos de un ambiente de nube real. Las tablas 8 y 9 muestran la información utilizada para realizar las simulaciones.

En los experimentos realizados, se pueden asignar 2 contenedores por cada servidor físico. Esta condición se puede generalizar y depende de las características de los recursos y los contenedores. Los contenedores no pueden migrar entre los recursos físicos, esta situación incrementa la violación de SLA ya que se necesita detener la ejecución de los trabajos dentro del contenedor.

Tabla 8. Los valores de simulación para los recursos físicos.

Recursos físicos	
Número de servidores	25
Núcleos	1
MIPS	1,000
Memoria	1,000
Consumo energético ocioso	86 W
Consumo energético máximo	180 W

Tabla 9. Los valores de simulación para los recursos virtuales.

Recursos virtuales	
Número de contenedores	50
Núcleos	1
MIPS	500
Memoria	500
Tipo	Sistema operativo

4.3 Análisis de la carga de trabajo

La evaluación del rendimiento de las estrategias de asignación demanda una minuciosa experimentación en entornos de pruebas. El uso de cargas de trabajo realistas que pueden representar las condiciones de despliegue es un elemento vital para recrear las condiciones de los experimentos a realizar. Nuestra carga de trabajo se basa en trazas reales de computación de alto rendimiento (HPC - High Performance Computing) de un archivo de cargas de trabajos paralelas (Feitelson, Tsafrir, and Krakov 2014) y grids de Grid Workload Archive (Delft 2013). Este tipo de cargas es adecuado para entornos de la nube, donde los contenedores de sistema operativo están destinados a ejecutar trabajos en Grids y HPC.

Se tomaron como base las cargas de trabajo de nueve archivos presentados en la Tabla 10 (Feitelson, Tsafrir, and Krakov 2014). Se agregaron dos valores a cada trabajo de las cargas mencionadas relativos a la capacidad de procesamiento requerida s_j y la prioridad ρ_j para cada uno de los trabajos. Estos valores se generaron de manera aleatoria siguiendo una distribución normal.

La Figura 13 muestra las características de las cargas de trabajo, detallando la distribución del trabajo por días. Para obtener valores estadísticos válidos, se considera una carga de trabajo de treinta días. Las cargas de trabajo presentan el número total de trabajos de tipos CI (uso intenso del CPU) y MI (uso intenso de la memoria).

Tabla 10. Cargas de trabajo empleadas para la simulación.

Abreviación	Sitio
DAS2	Universidad de Ámsterdam
DAS2	Universidad Tecnológica de Delft
DAS2	Universidad de Utrecht
DAS2	Universidad de Leiden
KHT	Instituto de tecnología Royal
DAS2	Universidad de Ámsterdam Vrije
HPC2N	Centro Norte de Supercómputo, Suecia
CTC	Centro para computación avanzada, Universidad de Cornell
LANL	Laboratorio Nacional de Los Álamos

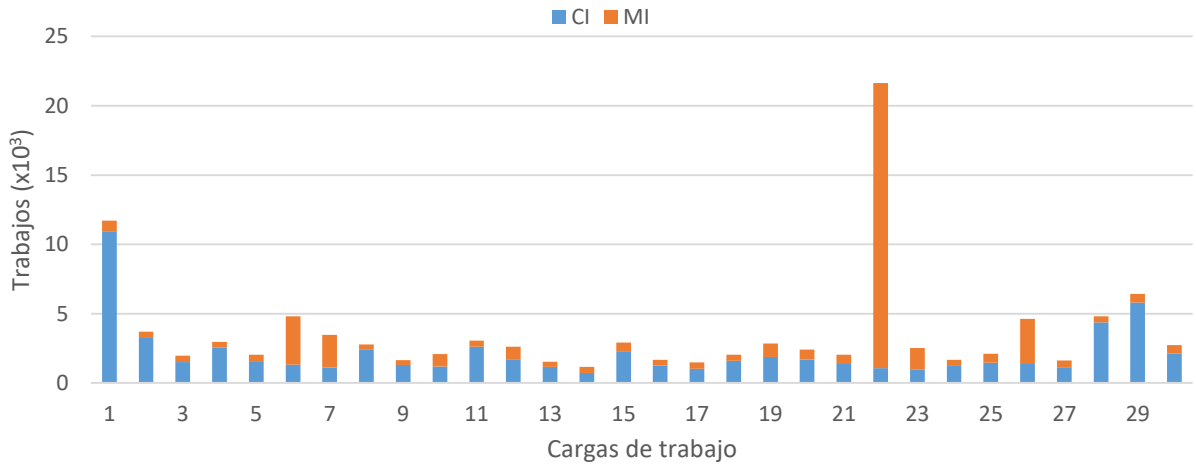

Figura 13. Distribución de los trabajos por días.

En la Figura 14 se muestra un histograma con la cantidad de trabajos por horas del día en la que llegan a la nube. Se puede observar que la mayoría de trabajos arriban entre las 6 AM y las 2 PM.

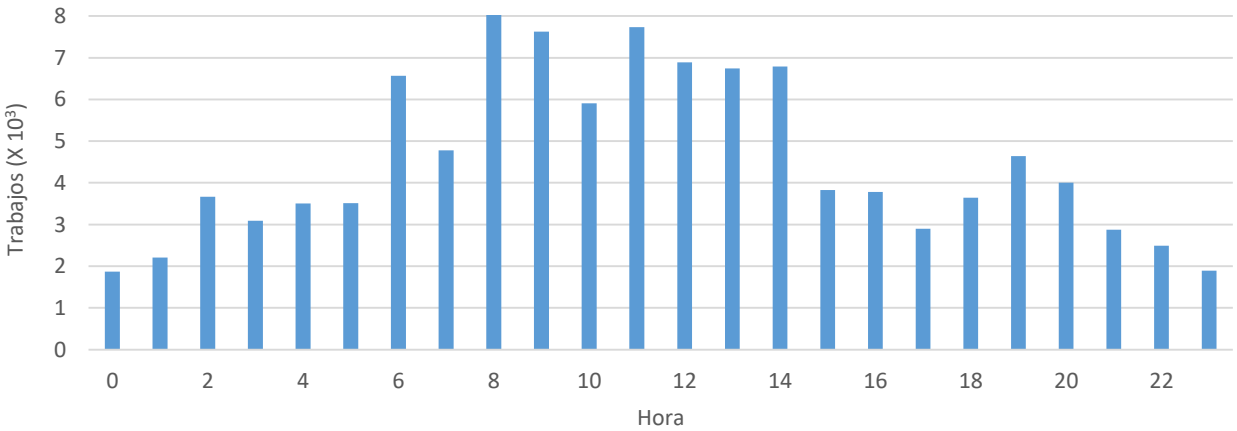

Figura 14. Distribución de tiempos de liberación de trabajos por hora.

4.4 Resultados

Primeramente, se muestran los resultados para el primer escenario en el cual se implementaron 58 variantes de estrategias, obtenidas a partir de las combinaciones de las 12 estrategias básicas y los 3 niveles. Con posterioridad, se muestran los resultados obtenidos para las 14 variantes de estrategias, implementadas a partir de las combinaciones de 7 estrategias y 2 niveles de asignación.

En ambos escenarios, la distribución de los recursos del procesador entre los trabajos es equitativa cuando la demanda de los trabajos asignados es mayor a la capacidad del contenedor. Es decir, se reduce la cantidad de CPU a todos los trabajos que se encuentran ejecutándose en el contenedor, de igual manera, disminuyendo su velocidad de procesamiento actual. La disminución de la velocidad se debe a la infraestructura de la nube, donde se cuenta con recursos limitados y estos deben proporcionar servicio a los usuarios, aunque no se tengan recursos suficientes para garantizar los SLA.

4.4.1. Modelo con capacidad requerida del contenedor

En este escenario, la capacidad de procesamiento de un trabajo en el CPU está limitada por la velocidad mínima requerida del trabajo. Es decir, los trabajos no reciben más recursos de CPU de los estipulados en el SLA. El objetivo principal es no saturar los recursos de cómputo y disminuir el consumo de energía de la infraestructura.

En el Anexo 1 se presentan las degradaciones promedio de C_{max}, SLA y Energía, calculadas a partir de los resultados obtenidos de la simulación de 30 días con sus respectivas cargas de trabajo. A medida que la degradación promedio es menor, mejor se considera la estrategia. La columna Media muestra el promedio de las tres columnas anteriores: C_{max}, SLA y Energía.

Las columnas Rango SLA, Rango C_{max}, y Rango Energía incluyen los rangos de cada una de las métricas. El rango para cada métrica se obtiene a partir del ordenamiento de las estrategias según sus resultados en orden ascendente, donde el rango 1 se corresponde con la estrategia de menor degradación promedio, es decir, la mejor.

La última columna, Rango Medio, se corresponde con el promedio de los rangos obtenidos por cada estrategia en cada uno de los criterios (las tres columnas anteriores).

En la Figura 15 se observa que la mejor estrategia para minimizar el SLA es BFit_MinTask_MinSla que asigna los trabajos basándose en la estrategia de mejor ajuste en el primer nivel. MintTask y MinSla

asignan las tareas al contenedor con menos tareas y menores violaciones SLA en el segundo y tercer nivel, respectivamente.

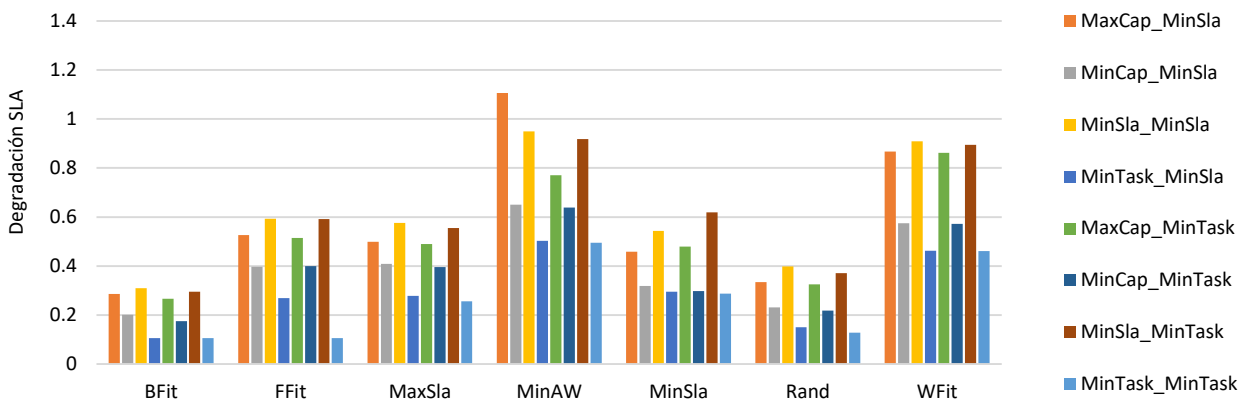

Figura 15. Degradación de SLA promedio por estrategias (escenario 1).

La estrategia con mejores resultados para la minimización del consumo de energía es BFit_MaxCap_MinTask. Esta estrategia realiza la asignación por mejor ajuste en el primer nivel. En el segundo y tercer nivel MaxCap y MinTask, respectivamente, tienen el mejor comportamiento de acuerdo a la degradación del desempeño. La Figura 16 muestra que la estrategia tiene los mejores resultados minimizando la energía (E).

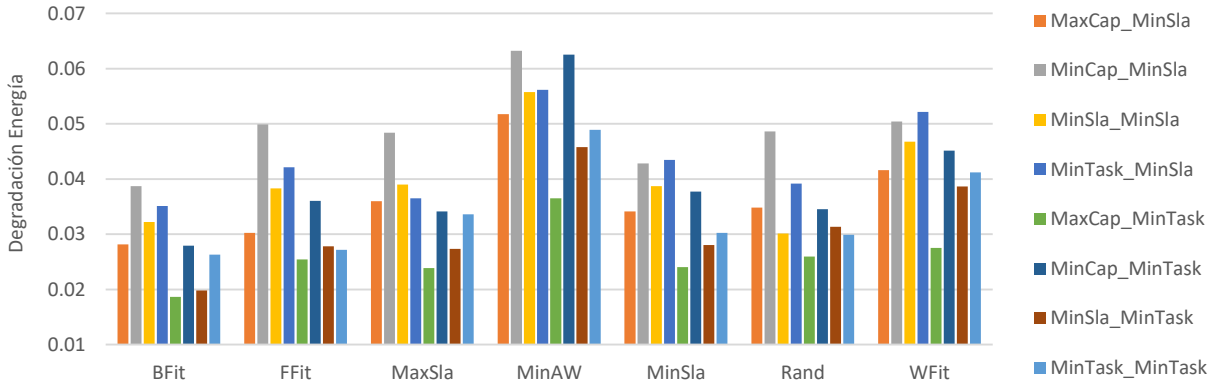

Figura 16. Degradación de energía promedio por estrategias (escenario 1).

Como se expone en la sección 3.3, tomar en consideración valores basados en el promedio puede propiciar interpretaciones erróneas. Una pequeña porción de los resultados obtenidos para una estrategia con una gran desviación podría cambiar de manera significativa los valores de la media. Para analizar los resultados, con mayor detalle se presentan los perfiles de desempeño de las estrategias implementadas.

Las mejores estrategias para minimizar el C_{max} son MaxSla_MaxCap_MinTask, MinAW_MaxCap_MinTask y Rand_MaxCap_MinTask (ver Figura 17). Estas estrategias asignan las tareas

al contenedor donde hay menor cantidad de trabajos cuando todos los contenedores se encuentran utilizando toda su capacidad de procesamiento (nivel 3).

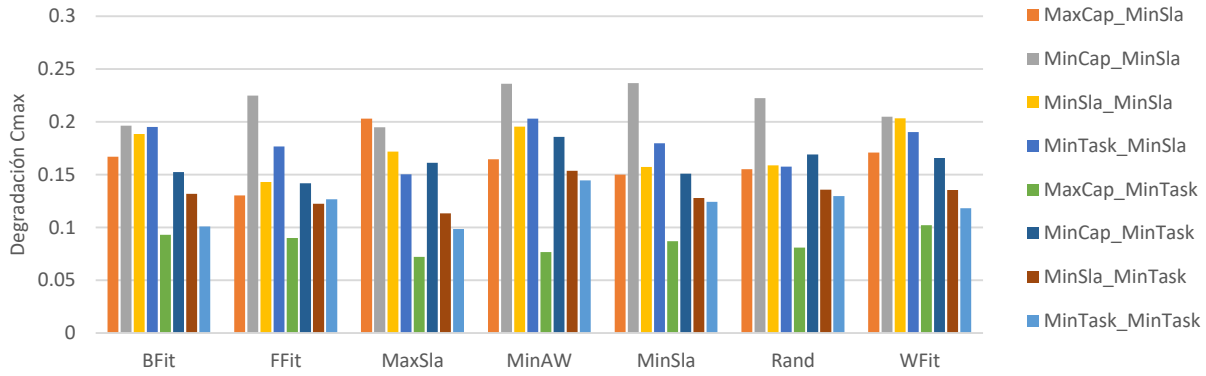

Figura 17. Degradación de $CMax$ promedio por estrategias (escenario 1).

La Figura 18 muestra los perfiles de desempeño de las 10 mejores estrategias de acuerdo a los valores de violaciones SLA en el intervalo $\tau = [1.0, 2.2]$. Se pueden apreciar discrepancias entre estos valores y los obtenidos con la métrica de degradación del desempeño en un porcentaje considerable.

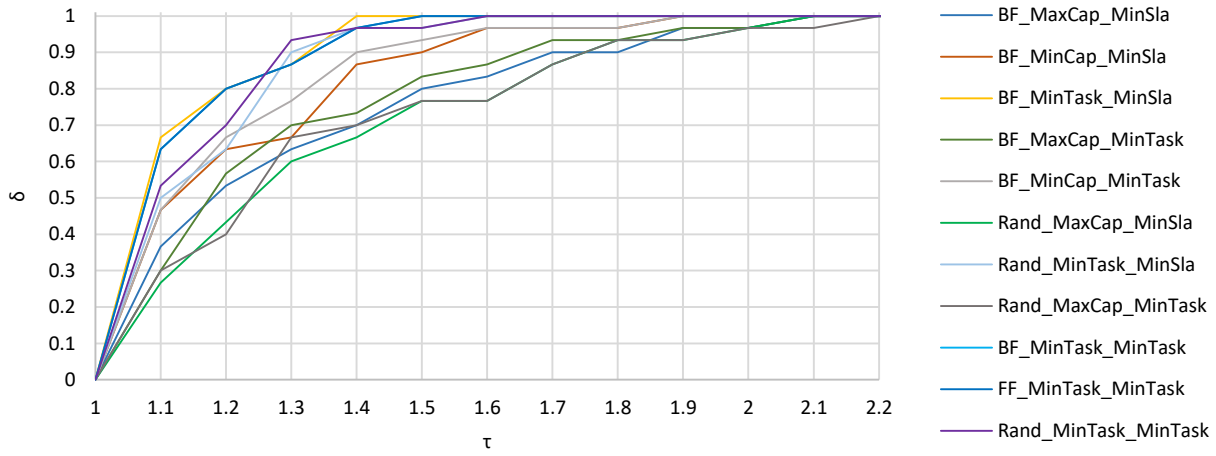

Figura 18. Perfil de desempeño de violaciones SLA para las mejores 10 estrategias.

La estrategia BFit_MinTask_MinSla tiene el mejor rango y la mayor probabilidad de ser la mejor estrategia. Si se elige estar dentro de un factor de 1.4 del mejor resultado como alcance de interés, la probabilidad de que esta estrategia obtenga los mejores resultados para el problema planteado es igual a 1. La estrategia FFit_MinTask_MinTask tiene el segundo rango más alto de ser la mejor estrategia. Con un factor de 1.4 como alcance de interés para nuestro problema con una probabilidad de 0.96.

La Figura 19 muestra los perfiles de desempeño relativos a la métrica de energía en el intervalo $\tau = [1,1.14]$. Esta ilustra las diferencias asociadas a la degradación de la energía para un porcentaje considerable de las soluciones obtenidas. La estrategia BFit_MaxCap_MinTask posee el rango y la

probabilidad más altos de ser la mejor estrategia. Si se desea seleccionar una estrategia dentro de un factor de 1.06, la probabilidad de que esta estrategia tenga el mejor desempeño es igual a 1. La estrategia WFit_MaxCap_MinTask es la segunda estrategia con mayor rango de convertirse en la mejor estrategia. Dentro un factor de 1.08 del alcance de nuestro interés, esta estrategia obtiene el mejor desempeño con una probabilidad de 1.

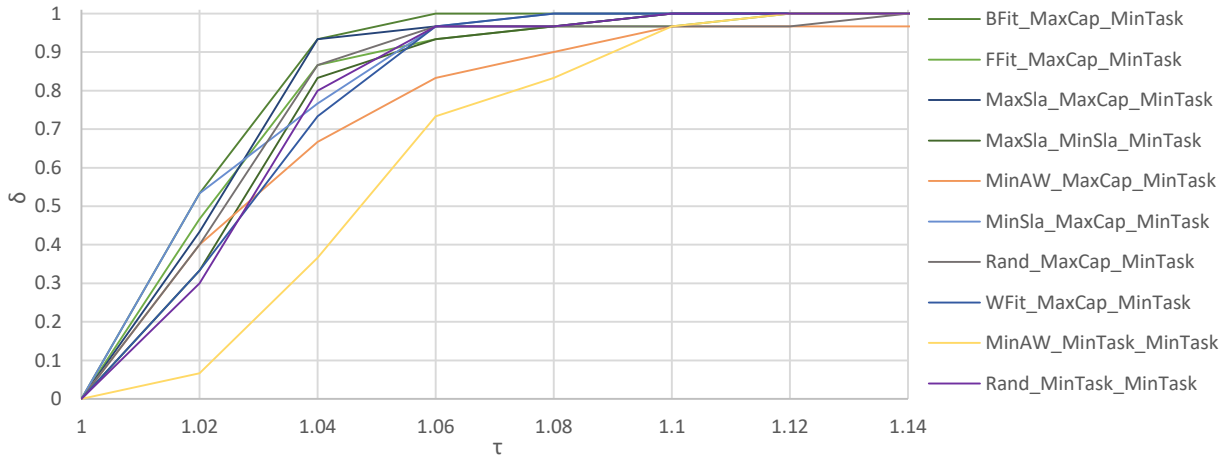

Figura 19. Perfil de desempeño de la energía para las mejores 10 estrategias.

En la Figura 20 muestra los perfiles de desempeño relativos al criterio de optimización C_{max} en el intervalo $\tau = [1,1.4]$. Esta ilustra las diferencias asociadas a la degradación del C_{max} para un porcentaje de las soluciones obtenidas. La estrategia con mejor desempeño para este criterio de optimización es: MinAW_MaxCap_MinTask si se selecciona dentro de un factor de 1.22, la probabilidad de que esta estrategia tenga el mejor desempeño es igual a 1.

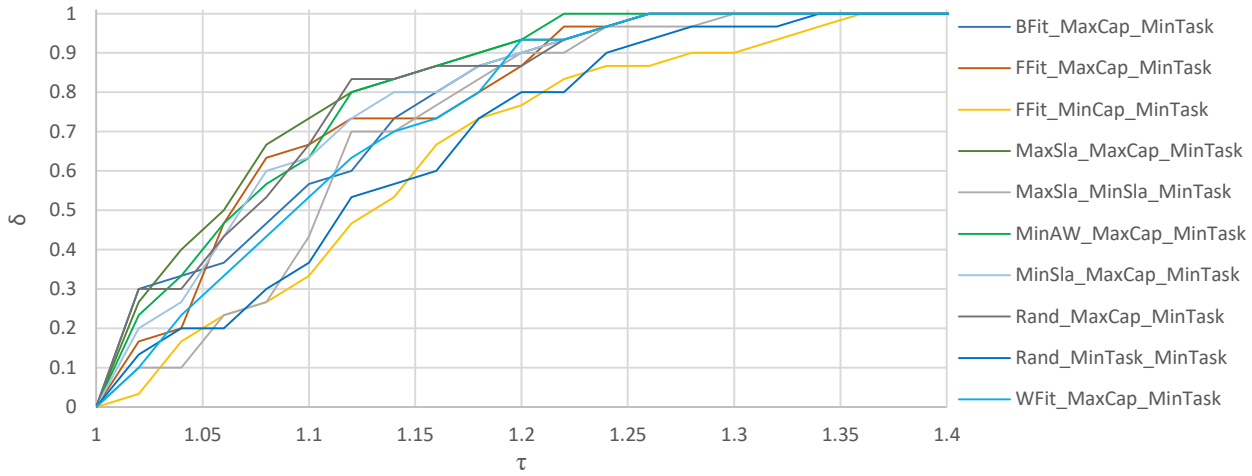

Figura 20. Perfil de desempeño para el criterio de optimización C_{max}.

La Figura 21 presenta los perfiles de desempeño de Random, Round Robin y las mejores 12 estrategias (14 estrategias en total) tomando como referencia el promedio de las métricas: C_{max}, violaciones SLA y E. El elemento más significativo de la Figura 20 es que en este subconjunto de estrategias BFit_MinTask_MinTask, FFit_MinTask_MinTask y Rand_MinTask_MinTask dominan al resto de las estrategias.

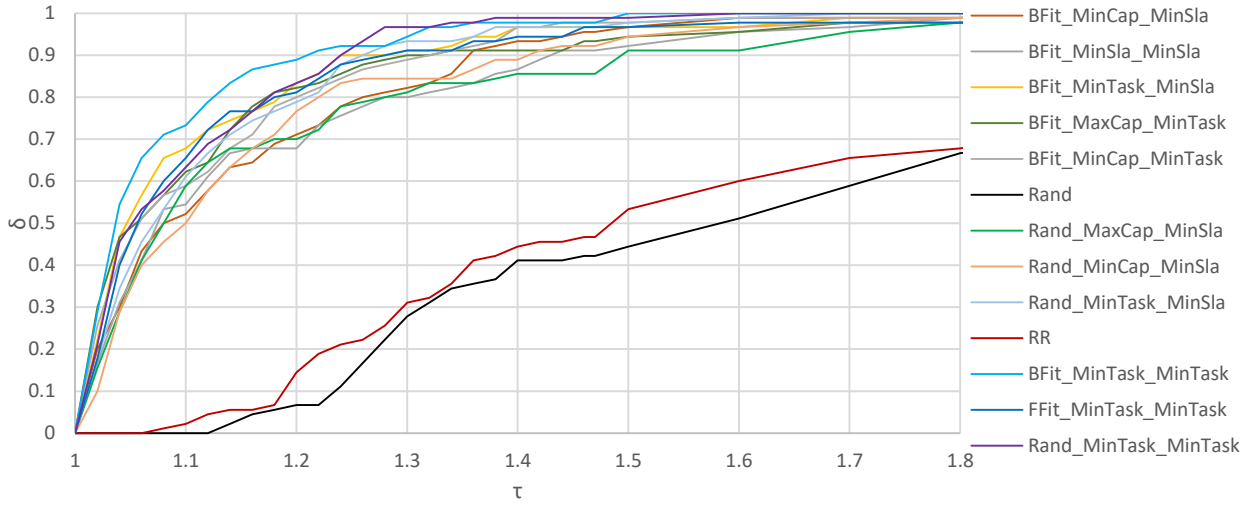

Figura 21. Perfil de desempeño promedio para violaciones SLA, C_{max} y E para 14 estrategias.

4.4.2. Modelo con capacidad máxima del contenedor.

En este segundo escenario, la capacidad de procesamiento de un trabajo en el CPU está limitada por la capacidad del contenedor donde se ejecuta el trabajo. Es decir, los trabajos pueden recibir más recursos de CPU de los estipulados en el SLA. La asignación de los recursos excedentes se realiza de forma proporcional a la capacidad mínima de cada trabajo. Es decir, cuando el contenedor satisface la capacidad mínima de todos los trabajos y sobra espacio en el CPU, entonces el espacio disponible es proporcional al

mínimo requerido por cada trabajo. El objetivo principal es maximizar la utilización de los recursos y disminuir el tiempo de procesamiento.

La Tabla 11 presenta al lector los resultados obtenidos para las 14 estrategias implementadas cuando se asigna toda la capacidad de procesamiento disponible a las tareas.

Tabla 11. Rangos y degradación media para estrategias en contenedores con modelo de capacidad máxima.

Estrategia	SLA	C_{max}	Energía	Media	Rango SLA	Rango C_{max}	Rango Energía	Rango Media
BFit_MinSla	0.754	0.120	0.036	0.304	7	11	11	7
BFit_MinTask	0.738	0.071	0.014	0.274	5	4	1	5
FFit_MinSla	13.938	0.197	0.042	4.726	14	14	12	14
FFit_MinTask	1.204	0.084	0.030	0.439	9	5	4	9
MaxSla_MinSla	1.259	0.130	0.032	0.474	10	12	6	10
MaxSla_MinTask	0.683	0.088	0.024	0.265	3	6	2	3
MinAW_MinSla	0.830	0.118	0.036	0.328	8	10	10	8
MinAW_MinTask	0.207	0.071	0.031	0.103	2	3	5	2
MinSla_MinSla	1.461	0.109	0.035	0.535	11	8	8	11
MinSla_MinTask	0.684	0.094	0.026	0.268	4	7	3	4
Rand_MinSla	2.697	0.151	0.075	0.974	13	13	14	13
Rand_MinTask	2.632	0.068	0.056	0.919	12	2	13	12
WFit_MinSla	0.754	0.112	0.036	0.300	6	9	9	6
WFit_MinTask	0.184	0.055	0.033	0.091	1	1	7	1
BFit_MinTask	0.738	0.071	0.014	0.274	5	4	1	5

En la Figura 22 se observa que la mejor estrategia para minimizar el SLA es WFit_MinTask, seguida por las estrategias MinAW_MinTask y MaxSla_MinTask. Para todas las estrategias implementadas en este escenario, los valores de degradación de SLA son menores para estrategias con MinTask que para MinSla.

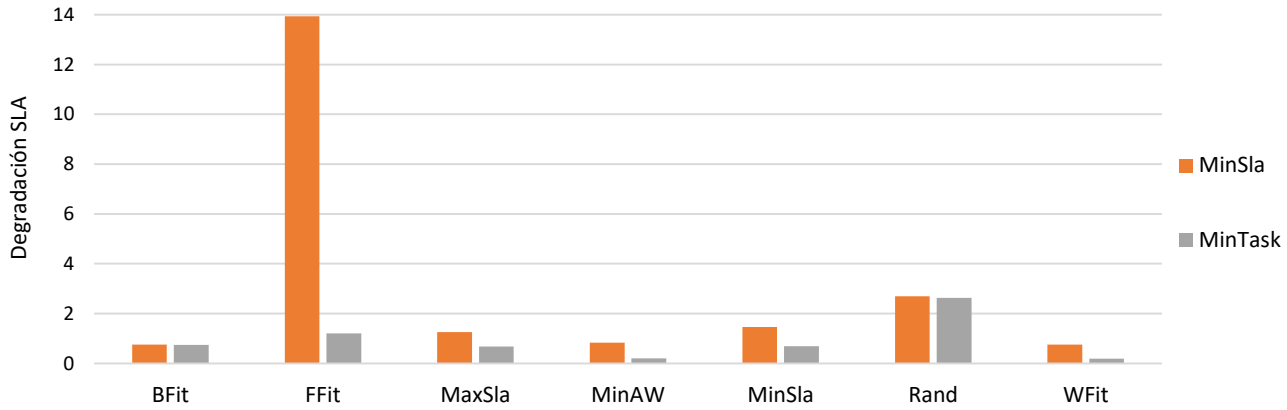

Figura 22. Degradación de SLA promedio por estrategias (escenario 2).

La estrategia con mejores resultados para la métrica de energía es BFit_MinTask. Nótese que MinTask, donde se asignan las tareas al contenedor con menor cantidad de trabajos, vuelve a tener los mejores resultados para esta métrica (ver Figura 23).

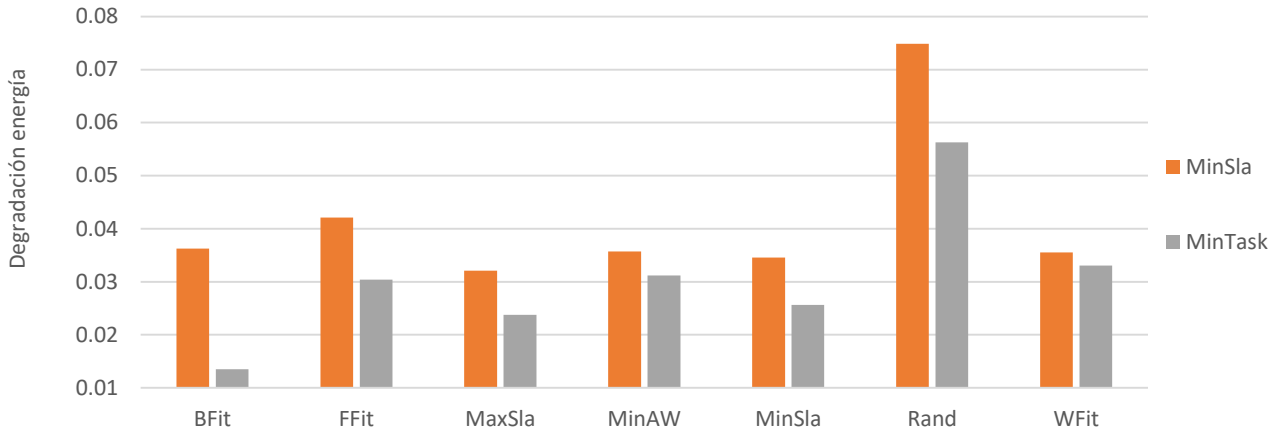

Figura 23. Degradación de energía promedio por estrategias (escenario 2).

En la Figura 24 se observa que la mejor estrategia para minimizar el C_{max} es WFit_MinTask que asigna los trabajos basándose en la estrategia de peor ajuste en el primer nivel. Al igual que en las otras dos métricas, MinTask obtiene los mejores resultados frente a MinSla cuando los contenedores tienen ocupada su capacidad máxima de procesamiento.

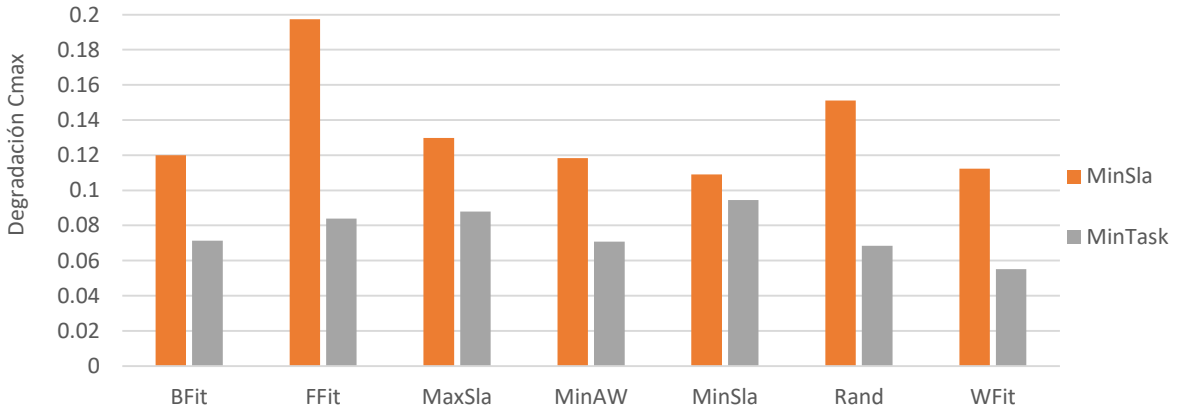

Figura 24. Degradación de C_{max} promedio por estrategias (escenario 2).

En la Figura 25, el perfil de desempeño muestra que la mejor estrategia para minimizar el SLA es WFit_MinTask seguida de la estrategia MinAW_MinTask.

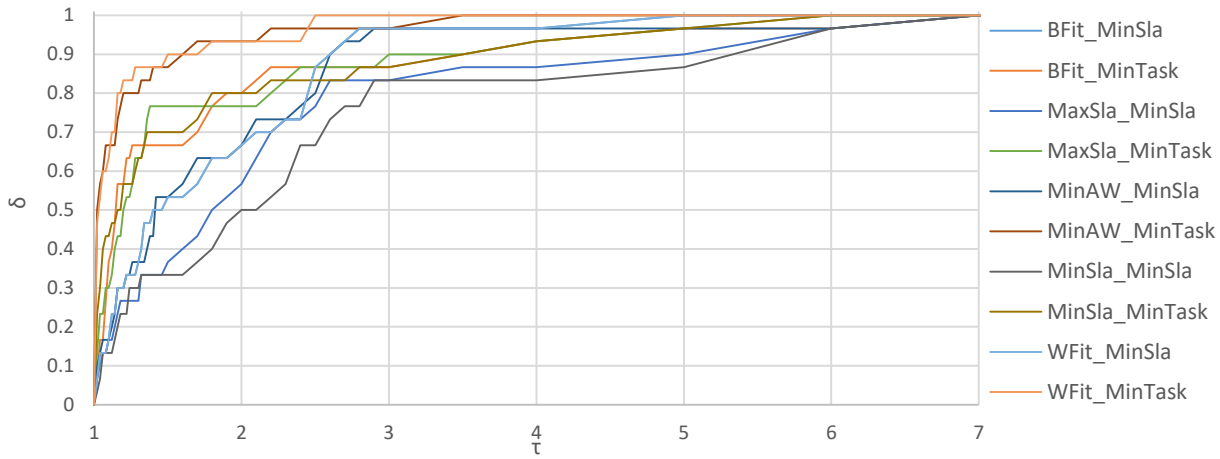

Figura 25. Perfil de desempeño para el criterio de optimización SLA.

La Figura 26 muestra que la estrategia con mejor perfil de desempeño para C_{max} es WFit_MinTask que asigna los trabajos basándose en la estrategia de peor ajuste en el primer nivel. MintTask asigna las tareas al contenedor con menos tareas.

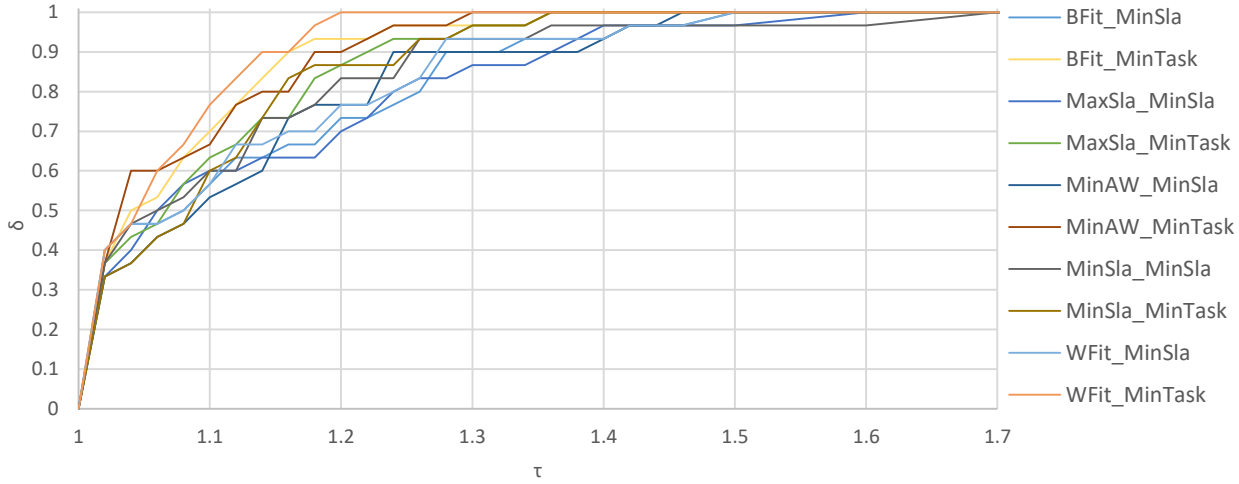

Figura 26. Perfil de desempeño para el criterio de optimización C_{max}.

En la Figura 27 se muestran las 10 mejores estrategias para la métrica E de acuerdo a su perfil de desempeño. Si se elige un factor de 1.08 del mejor resultado como alcance de interés, el porcentaje de soluciones de FFit_MinTask, que ganan en un problema determinado es del 100%.

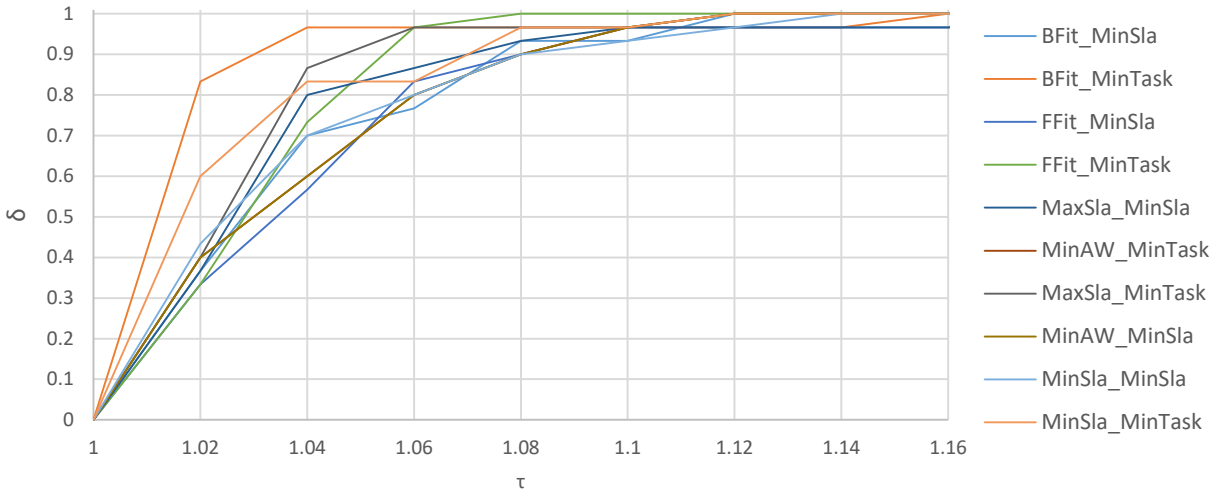

Figura 27. Perfil de desempeño para el criterio de optimización energía (E).

Se presentan al lector los perfiles de desempeño de las 14 estrategias implementadas para el segundo escenario, donde se utiliza el promedio de las tres métricas: SLA, C_{max} y energía. Si se selecciona un factor de 2.2, el porcentaje de las estrategias MinAW_MinTask y WFit_MinTask que ganan en un problema determinado es de 97% y 95 %, respectivamente (ver Figura 28).

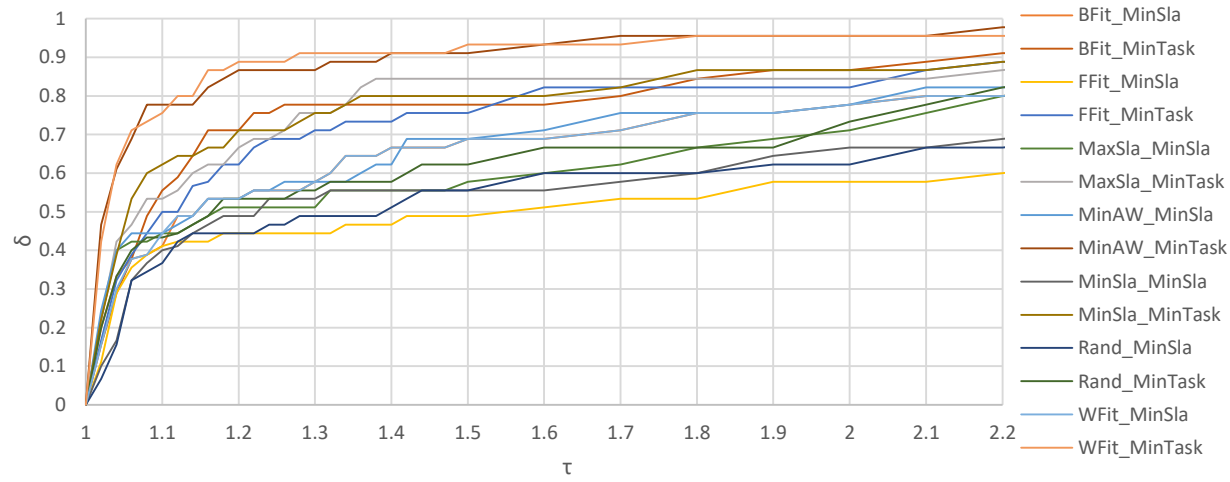

Figura 28. Perfil de desempeño promedio para violaciones SLA, C_{max} y E para 14 estrategias.

En este capítulo de se presentaron los detalles de configuración del sistema utilizado, las características de las cargas de trabajo empleadas y los resultados obtenidos. En el siguiente capítulo se realiza la discusión de los resultados obtenidos como parte de la presente tesis.

Capítulo 5. Discusión

La sección 4.4 presenta los resultados para los dos escenarios implementados. Para el caso del primer escenario, se definieron 3 niveles que conforman el mecanismo de asignación de las tareas a los contenedores. En este escenario, se implementan 58 variantes de estrategias, obtenidas a partir de las combinaciones de las 7 estrategias básicas en el primer nivel (BFit, FFit, MaxSla, MinAW, MinSla, Rand y WFit), 4 en el segundo (MaxCap, MinCap, MinSla y MinTask) y 2 en el tercer nivel(MinSla y MinTask) (7*4*2=56) y adicionalmente las estrategias Round Robin y Rand, siguiendo el modelo con capacidad requerida descrito en la subsección 3.4.1. En cada una de ellas se obtienen los promedios de las degradaciones de las tres métricas para los 30 días: C_{max} , violaciones de SLA y energía. En el primer escenario, la capacidad de procesamiento de un trabajo en el CPU no puede exceder a la capacidad de procesamiento requerida del trabajo definida en el SLA.

Se presentan para el segundo escenario los resultados de las 14 variantes de estrategias, obtenidas a partir de las combinaciones de las 7 estrategias básicas del primer nivel (BFit, FFit, MaxSla, MinAW, MinSla, Rand y WFit) y 2 del segundo nivel (MinSla y MinTask) siguiendo el modelo con capacidad completa descrito en la subsección 3.4.2. Este escenario asigna toda la capacidad de procesamiento disponible en el contenedor a las tareas que se encuentran en ejecución. Por tal motivo, solo existen dos niveles relativos al proceso de asignación de tareas al contenedor. El primero se refiere a cuando existen contenedores con la capacidad de procesamiento requerida por la tarea (cuando se considera el mínimo definido por el SLA para cada trabajo dentro de un contenedor) y el segundo cuando ningún contenedor cuenta con la capacidad suficiente para ejecutar el trabajo.

5.1. Rangos y degradación media para las mejores estrategias

Para las 72 estrategias implementadas se obtuvieron los rangos de cada de las métricas, así como el promedio de estos rangos (ver Anexo 2). A continuación, se seleccionaron las estrategias que obtuvieron los rangos 1, 2 o 3 en algunas de las métricas, así como las dos con peor rango. Los resultados de esta selección se muestran en la Tabla 12, donde se resaltan la mejor estrategia en cada criterio.

Según la Tabla 12, la estrategia WFit_MinTask correspondiente al segundo escenario resulta la de mejor desempeño para las violaciones SLA y la energía, con una degradación promedio de 0.185 y 0.081, respectivamente. Esto significa que, si existen contenedores con capacidad disponible, calculada como la diferencia entre la capacidad del contenedor y la suma de las capacidades de procesamiento mínimas requeridas por cada trabajo, más la capacidad de procesamiento mínima requerida por el trabajo que se

desea asignar, se debe escoger aquel que posea la mayor capacidad disponible. En el caso que no se diera la situación anterior, se debe escoger aquel contenedor donde se ejecuta la menor de cantidad de tareas.

Tabla 12. Comparativa entre estrategias de los dos escenarios.

Estrategia	Escenario	SLA	C_{max}	Energía	Media	Rango SLA	Rango C_{max}	Rango Energía	Rango Media
BFit_MinTask	2	0.738	**0.025**	0.097	0.287	5	**1**	6	5
MaxSla_MaxCap_MinTask	1	11.752	0.050	0.090	3.965	41	15	2	39
MaxSla_MinTask	2	0.683	0.035	0.114	0.278	3	2	12	3
MinAW_MinTask	2	0.208	0.043	0.097	0.116	2	5	5	2
MinSla_MinTask	2	0.685	0.034	0.121	0.281	4	3	15	4
Rand	1	129.48	0.295	0.827	43.53	72	72	72	72
Rand_MinTask	2	2.632	0.068	0.094	0.932	12	50	3	12
RR	1	122.29	0.274	0.715	41.09	71	71	71	71
WFit_MinTask	2	**0.185**	0.044	**0.081**	0.104	**1**	7	**1**	1

La mejor estrategia, considerando el criterio C_{max}, es BFit_MinTask con una degradación promedio de 0.025. Esto significa que, si existen contenedores con capacidad disponible, calculada como la diferencia entre la capacidad del contenedor y la suma de las capacidades de procesamiento mínimas requeridas por cada trabajo, más la capacidad de procesamiento mínima requerida por el trabajo que se desea asignar, se debe escoger aquel que posea la menor capacidad disponible. En el caso que no se diera la situación anterior, se debe escoger aquel contenedor donde se ejecuta la menor de cantidad de tareas.

Como se puede apreciar, las mejores estrategias según las degradaciones promedio son las estrategias del escenario 2. Sin embargo, se puede apreciar que la estrategia MaxSla_MaxCap_MinTask del escenario 1 tiene un buen desempeño con respecto al criterio de la energía, comparable con la estrategia WFit_MinTask del escenario 2, lo que se aprecia en la pequeña diferencia entre los promedios de sus degradaciones, 0.090 y 0.081, respectivamente. Nótese que el buen desempeño en cuanto al criterio de energía contrasta con un bajo desempeño ya que en el primer nivel la estrategia MaxSla_MaxCap_MinTask prioriza la selección de los contenedores con más violaciones SLA. Esto evidencia que los criterios de energía y violaciones SLA se contraponen en el escenario 1.

La estrategia aleatoria es la estrategia con peor desempeño con respecto a la degradación de todas las métricas. Round Robin, la penúltima estrategia de acuerdo a Rango, es una estrategia ampliamente utilizada para distribuir la carga por su facilidad de implementación y la utilización de poca información para tomar decisiones. Las 70 estrategias propuestas pueden mejorar el desempeño que ofrece Round Robin logra.

5.2. Perfil de desempeño para las mejores estrategias

A continuación, se presenta el perfil de desempeño de las nueve estrategias relacionadas en la Tabla 12. La Figura 29 muestra que la mejor estrategia tomando como referencia el promedio de los perfiles de desempeño es WFit_MinTask con un 100% de obtener la mejor solución si se selecciona un factor de 2.5.

La estrategia MinAW_MinTask también tiene un buen desempeño ya que logra que se obtenga la mejor solución en el 98.8% de los casos si se selecciona el mismo factor. Un elemento importante de este análisis radica en que, en la mayoría de las estrategias cuando los contenedores se encuentran a su máxima capacidad, es preferible asignar la tarea al contenedor con menos tareas que a donde existan menos violaciones de SLA. Adicionalmente, la mayor parte de las estrategias que integran este grupo pertenecen al segundo escenario.

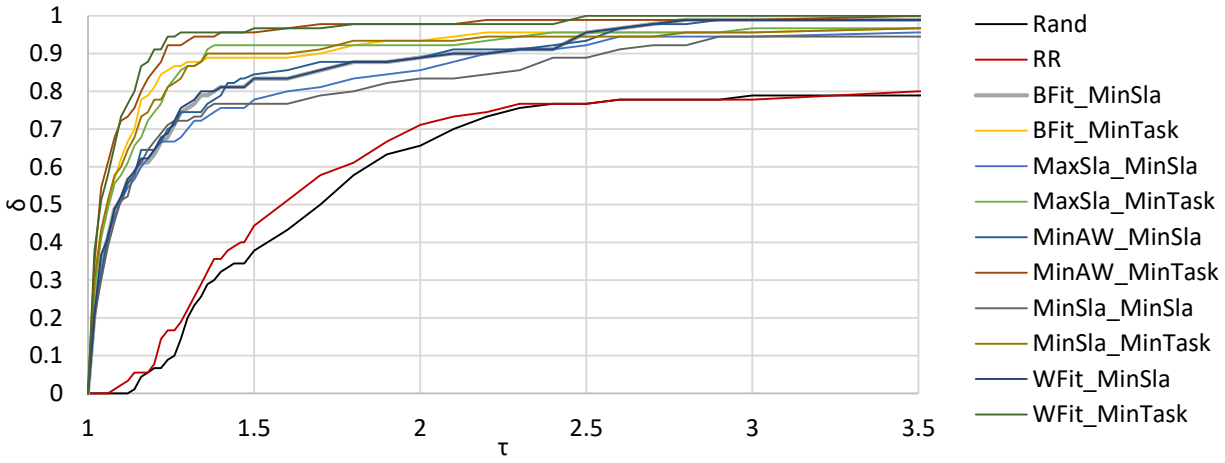

Figura 29. Perfiles de desempeño para las mejores estrategias.

Capítulo 6. Conclusiones

El análisis presentado en este trabajo y las publicaciones recientes que abordan problemas similares, permitieron establecer las principales tendencias en la utilización de contenedores en la computación en la nube.

Como resultado del estudio realizado, se propusieron 12 estrategias básicas agrupadas en tres niveles para la asignación de trabajos a un conjunto C de m contenedores que se ejecutan en un conjunto P de M servidores. Las estrategias consideran la minimización del tiempo de terminación de todas las tareas (C_{max}), las violaciones de SLA y la energía del centro de datos.

Empleando el simulador CloudSim, se implementaron 72 variantes de estrategias. Para el escenario de capacidad requerida se implementaron 58 y para el escenario de capacidad completa 14 estrategias.

Se definieron 2 escenarios que permitieron validar las estrategias propuestas en el simulador CloudSim.

Los resultados del análisis experimental comparativo realizado, tomando como base la degradación del desempeño, permitieron establecer que:

- Según la degradación del desempeño, la mejor estrategia para minimizar el SLA es BFit_MinTask_MinSla.
- Según la degradación del desempeño, la estrategia con las que se obtienen mejores resultados para el objetivo Energía es BFit_MaxCap_MinTask.
- Según la degradación del desempeño, las mejores estrategias para la métrica C_{max} son MaxSla_MaxCap_MinTask, MinAW_MaxCap_MinTask y Rand_MaxCap_MinTask.

Los resultados del análisis experimental, tomando como base los perfiles de desempeño, permiten plantear que:

- De acuerdo a las violaciones de SLA, la estrategia BFit_MinTask_MinSla tiene el mejor rango y la mayor probabilidad de ser la mejor estrategia.

- Según los perfiles de desempeño relativos a la métrica de energía, la estrategia BFit_MaxCap_MinTask posee el rango y la probabilidad más alta de ser la mejor estrategia.

- Según los perfiles de desempeño relativos al criterio de optimización C_{max}, la estrategia que mejor desempeño muestra es: MinAW_MaxCap_MinTask.

Los resultados del análisis de los perfiles de desempeño tomando como referencia el promedio de las métricas: C_{max} , violaciones de SLA y Energía permitieron establecer que las estrategias BFit_MinTask_MinTask, Rand_MinTask_MinTask y Rand_MinTask_MinSla dominan al resto.

Para el segundo escenario, donde a los trabajos se les puede asignar más capacidad de procesamiento que la requerida, los resultados del análisis experimental realizado, tomando como base la degradación del desempeño, permitieron establecer que:

- Según la degradación del desempeño, la mejor estrategia para minimizar el SLA es WFit_MinTask.

- El mejor valor de energía en cuanto a la degradación de desempeño se obtiene con la estrategia BFit_MinTask.

- El mejor resultado para el C_{max}, según la degradación del desempeño se obtiene con la estrategia WFit_MinTask.

Los resultados del análisis experimental, tomando como referencia los perfiles de desempeño, permiten plantear que:

- De acuerdo a las violaciones de SLA, la estrategia, WFit_MinTask tiene el mejor rango y la mayor probabilidad de ser la mejor estrategia.

- Según los perfiles de desempeño relativos al criterio de optimización C_{max}, la estrategia que mejor desempeño muestra es WFit_MinTask.

- Según los perfiles de desempeño relativos a la métrica de energía, la estrategia FFit_MinTask posee el rango y la probabilidad más alta de ser la mejor estrategia.

Los resultados del análisis de los perfiles de desempeño tomando como referencia el promedio de las métricas: C_{max}, violaciones SLA y Energía permitieron establecer que las estrategias WFit_MinTask y MinAW_MinTask dominan al resto.

Si se realiza un análisis comparativo entre las estrategias de ambos escenarios (ver Capítulo 5), es posible afirmar que existen un número mayor de estrategias del segundo escenario que obtienen mejores resultados que la mayor parte de las correspondientes al primer escenario.

6.1 Contribución al conocimiento

La investigación contribuye al problema de asignación de trabajos en contenedores para el modelo de cómputo en la nube, las aportaciones de este trabajo de tesis son:

- Se desarrolla un marco de simulación para sistemas nube basados en contenedores.
- Se aborda el problema de asignación de trabajos considerando tres criterios en sistemas de nube basados en contenedores.
- Se proponen dos modelos de procesamiento de trabajos en nubes computacionales basados en nubes.
- Se analizan estrategias de asignación con dos y tres niveles de asignación.
- Se desarrolla un marco de simulación para sistemas nube basados en contenedores.

Estas aportaciones son la base de las siguientes presentaciones:

6.1.1 Conferencias

1. Andrei Tchernykh, Rewer Canosa, Jorge M. Cortés-Mendoza, Arutyun Avetisyan, Raul Rivera-Rodriguez. Bi-objective strategies for resource optimization in containerized cloud computing. CARLA 2018 - Latin American Conference on High Performance Computing. Piedecuesta, Colombia September 26th – 28th 2018. The special procceedings of the conference. Short paper (accepted).
2. Rewer Canosa, Andrei Tchernykh. Bi-objective optimization strategies for resource management in container-based clouds. ISUM 2018 - 9th International Supercomputing Conference in Mexico. Mérida, Yucatán, México. March 5 - 9 2018.

3. Rewer Canosa, Andrei Tchernykh. Bi-objective optimization strategies for resource management in container-based clouds. Club Temático Linux. Unión de Informáticos de Cuba. Cienfuegos, Cuba. Diciembre 8, 2017.

6.2 Limitaciones de la investigación

La siguiente sección describe las limitaciones de esta investigación, éstas se dividen en tres principales aspectos: el ambiente nube, la carga de trabajo y los modelos de asignación.

El ambiente de simulación es una abstracción limitada de un ambiente real de nube donde los aspectos más representativos son modelados. El enfoque contempla elementos relevantes que afectan significativamente el costo y la calidad del servicio, pero omite otros menos importantes. El modelo propuesto es representativo de entornos nubes reales y puede ser la base de futuros trabajos donde pueden ser agregados componentes adicionales.

Las cargas utilizadas son una mezcla de trabajos de diferentes centros de supercómputo (ver Tabla 9). Las limitaciones con respecto a la carga de trabajo incluyen: no poseen valores para reflejar la capacidad de procesamiento mínima requerida, los tamaños de las cargas de trabajo utilizadas varían considerablemente y las cargas de trabajo no poseen información sobre las prioridades de ejecución y terminación de los trabajos.

Los modelos de asignación estiman la energía solamente para un sistema de servidores homogéneos del tipo utilizado. Una vez que los contenedores se encuentran utilizando toda su capacidad de procesamiento la distribución de esta se realiza únicamente de forma equitativa sin atender parámetros como: cantidad de instrucciones pendientes a ejecutar, prioridad de los trabajos, etc.

6.3 Trabajo futuro

Después de realizar el análisis correspondiente para cada uno de los escenarios y estrategias implementadas, se considera el siguiente trabajo a futuro:

- Evaluar las estrategias presentadas de manera experimental para observar su comportamiento con cargas utilizadas en entornos nube reales basado en contenedores.

- Considerar la aplicación de nuevas estrategias de asignación en un escenario donde los trabajos tengan prioridades al redistribuir la capacidad de procesamiento del contenedor donde se ubique.

- Ampliar el estudio realizado a la asignación de tareas como microservicios en entornos CaaS.
- Mejorar la abstracción del entorno de simulación, varios elementos del entorno nube y ambientes CaaS no se consideran en esta investigación.

Literatura citada

Ahmad, N. 2017. Cloud computing: technology, security issues and solutions. 2nd International conference on anti-cybercrimes (ICACC): 30-35. DOI: 10.1109/Anti-Cybercrime.2017.7905258.

Al-Ghuwairi, A.R., Salah Z., Alsarhan, A., Al Qudah, S., Al Qahmous, G., Baarah, A., Al-Oqaily, A. 2018. Monitoring and modelling service level agreement of multiple virtual machines in cloud computing. International journal of business information systems 27(4): 538–553. DOI: 10.1504/IJBIS.2018.090293.

Alhamad, M., Dillon, T., Chang, E. 2010. Conceptual SLA framework for cloud computing. 4th IEEE international conference on digital ecosystems and technologies - conference proceedings of IEEE-DEST 2010 606–610. DOI: 10.1109/DEST.2010.5610586.

Alhamad, M., Dillon, T., Chang, E. 2010. SLA-based trust model for cloud computing. 13th international conference on network-based information systems: 321–324. DOI: 10.1109/NBiS.2010.67.

Armenta-Cano, F. A., A. Tchernykh, J. M. Cortes-Mendoza, R. Yahyapour, A. Yu. Drozdov, P. Bouvry, D. Kliazovich, A. Avetisyan, and S. Nesmachnow. 2017. Min_c: Heterogeneous concentration policy for energy-aware scheduling of jobs with resource contention. Programming and computer software 43(3): 204–215. Disponible en: http://link.springer.com/10.1134/S0361768817030021.

Armenta Cano, F. A., Chernykh, A. 2018. Modelo de energía basado en la concentración de tareas heterogéneas y diseño de estrategias de consolidación en cómputo en la nube. Tesis de Doctorado en Ciencias. Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California. 97 pp.

Badger, L., Grance, T., Patt-Corner, R., Voas, J.2012. Cloud computing synopsis and recommendations. National institute of standars and technology, special publication 800-146. Disponible en: http://csrc.nist.gov/publications/nistpubs/index.html.

Balasubramanian Sekar, V., Patil, V., Giusti, M., Bhide, A., Gupta, A. 2017. AWS EC2 vs. Joyent's triton: A comparison of docker container-hosting platforms. Proceedings of the 8th workshop on scientific cloud computing (ScienceCloud '17). ACM, New York, NY, USA, 33-36. DOI: 10.1145/3086567.3086572.

Barik, R., Lenka, R., Rao, R. 2016. Performance analysis of virtual machines and containers in cloud computing. International conference on computing, communication and automation (ICCCA). 1204–1210. DOI: 10.1109/CCAA.2016.7813925.

Barroso, L. A., Clidaras, J., Hölzle, U. 2009. The datacenter as a computer: An introduction to the design of warehouse-scale machines (1st ed.). Morgan and claypool publishers.

Basmadjian, Robert et al. 2011. A methodology to predict the power consumption of servers in data centres. Proceedings of the 2nd international conference on energy-efficient computing and networking: 1-10. ACM. DOI: 10.1145/2318716.2318718.

Beloglazov, A. 2013. Energy-efficient management of virtual machines in data centers for cloud computing. doctor of philosophy thesis. Department of computing and information systems, The University of Melbourne. 232 pp.

Beloglazov, A., Buyya, R. 2010. Energy efficient resource management in virtualized cloud data centers. 10th IEEE/ACM international conference on cluster, cloud and grid computing. 826-831. DOI: 10.1109/CCGRID.2010.46.

Beloglazov, A., Buyya, R. 2012. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurrency and computation: practice & experience 24(13): 1397-1420.DOI: 10.1002/cpe.1867.

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., Buyya, R. 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software-practice & experience 41(1): 23-50. DOI: 10.1002/spe.995.

Celesti, A., Mulfari, D., Fazio M. 2016. Exploring container virtualization in IoT clouds. IEEE international conference on smart computing (SMARTCOMP): 1-6.DOI: 10.1109/SMARTCOMP.2016.7501691

Cortés Mendoza, J. M., Chernykh, A. 2018. Optimización de costos y calidad de servicio en sistemas de voz sobre IP basados en nubes computacionales. Tesis de Doctorado en Ciencias. Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California. 144 pp.

Dhiman, G., Mihic, K., Rosing, T. 2010. A system for online power prediction in virtualized environments using gaussian mixture models. Design automation conference: 807-812.DOI: 10.1145/1837274.1837478.

DolanJorge, E.D., Moré, J. 2002. Benchmarking optimization software with performance profiles. Mathematical programming journal: 201–213.DOI: 10.1007/s101070100263.

Dua, R., Raja, A. R., Kakadia, D. 2014. Virtualization vs containerization to support PaaS. IEEE international conference on cloud engineering, 610 - 614. DOI: 10.1109/IC2E.2014.41.

Fan, X., Weber, W., Barroso, L.A. 2011. Power provisioning for a warehouse-sized computer. ACM SIGARCH computer architecture news 35(2): 13-23. DOI: 10.1145/1273440.1250665.

Feitelson, D. G., Tsafrir D., Krakov D. 2014. Experience with using the parallel workloads archive. Journal of parallel and distributed computing 74(10): 2967-2982. DOI: 10.1016/j.jpdc.2014.06.013.

Fotuhi Piraghaj, S. 2016. Energy-efficient management of resources in Container-based clouds. Doctor of Philosophy Thesis. Department of Computing and Information Systems, The University of Melbourne. 220 pp.

Fotuhi Piraghaj, S., Vahid Dastjerdi, A., Calheiros, R. N., Buyya, R. 2015. A framework and algorithm for energy efficient container consolidation in cloud data centers. IEEE international conference on data science and data intensive systems: 368 - 375. DOI: 10.1109/DSDIS.2015.67.

Fotuhi Piraghaj, S., Vahid Dastjerdi, A., Calheiros, R. N., Buyya, R. 2016. ContainerCloudSim: An environment for modeling and simulation of containers in cloud data centers. Journal of software: Practice and experience 47(4): 505-521. DOI: 10.1109/DSDIS.2015.67.

Garg S.K., Gopalaiyengar S.K., Buyya R. 2011. SLA-based resource provisioning for heterogeneous workloads in a virtualized cloud datacenter. Algorithms and architectures for parallel processing. ICA3PP 2011. Lecture notes in computer science, vol 7016. Springer, Berlin, Heidelberg: 371-384. DOI: 10.1007/978-3-642-24650-0_32.

Google. 2014. An update on container support on google cloud platform. Google cloud platform blog. Recuperado: el 15 de mayo de 2018 de: https://opensource.googleblog.com/2014/06/an-update-on-container-support-on.html

Grance, T., Mell, P.2012. Cloud computing synopsis and recommendations. National institute of standars and technology, special publication 800-145. Disponible en: http://csrc.nist.gov/publications/nistpubs/index.html.

Gupta, K. S., Banerjee, A., Abbasi, Z., Mukherjee, T., Varsamopoulos, G., Jonas, M., Ferguson J, Robin Gilbert, R., Mukherjee, T. 2014. GDCSim: A simulator for green data center design and analysis. ACM transactions on modeling and computer simulation (TOMACS). 24. 10.1145/2553083.

Hindman, B., Konwinski, A., Zaharia, M., Ghodsi, A., Joseph, A. D., Katz, R., Shenker, S., Stoica, I. 2011. Mesos: a platform for fine-grained resource sharing in the data center. Proceedings of the 8th USENIX conference on networked systems design and implementation (NSDI'11). USENIX Association, Berkeley, CA, USA: 295-308.

Hintemann, R., Beucker, S., Clausen, J., Stobbe, L., Proske, M., Nissen, N. F. 2016. Energy efficiency of data centers - A system-oriented analysis of current development trends. Electronics goes green 2016+ (EGG): 1-5. DOI: 10.1109/EGG.2016.7829805.

Kaneria, O., Banyal, R. K. 2016. Analysis and improvement of load balancing in cloud computing. international conference on ICT in business industry & government (ICTBIG): 1-5. DOI: 10.1109/ICTBIG.2016.7892711.

Karle, Akshay. 2015. Operating system containers vs. application containers. Recuperado: el 15 de mayo de 2018 de: https://blog.risingstack.com/operating-system-containers-vs-application-containers/

Kokane, M., Jain, P., Sarangdhar, P. 2013. Data storage security in cloud computing. International journal of advanced research in computer and communication engineering 2(3): 1388–1393.

Kumar Bansal, A., Gupta, V., Kaur, H. 2016. An implementation of private cloud's service model (IaaS) using lightweight virtualization. International conference on electrical, electronics, and optimization techniques, ICEEOT 2016: 950–954. DOI: 10.1109/ICEEOT.2016.7754826.

Kusic, D., Kephart, J. O., Hanson, J. E. 2008. Power and performance management of virtualized computing environments via lookahead control. International conference on autonomic Computing: 3 - 12. DOI: 10.1109/ICAC.2008.31.

Leitner, P., Ferner, Jo., Hummer, W., Dustdar, S. 2013. Data-driven and automated prediction of service level agreement violations in service compositions. Springer science+business media New York 2013: 447 - 470. DOI: 10.1007/s10619-013-7125-7.

Lin, C.T. 2013. Comparative Based Analysis of Scheduling Algorithms for Resource Management in Cloud Computing Environment. International Journal of Computer Science and Engineering 1(1): 17-23.

Liu, X.-F., Zhan, Z.-H., Zhang, J. 2017. An Energy Aware Unified Ant Colony System for Dynamic Virtual Machine Placement in Cloud Computing. Energies 10(5), 609. DOI: 10.3390/en10050609.

Lozano Guzmán, L. M., Chernykh, A. 2013. Análisis experimental de estrategias de calendarización en la nube considerando calidad de servicio y eficiencia energética. Tesis de Maestría en Ciencias. Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California. 111 pp.

Lyu, H. Li, P. Yan, R. 2016. Load forecast of resource scheduler in cloud architecture. International Conference on Progress in Informatics and Computing (PIC), 508 - 512. DOI: 10.1109/PIC.2016.7949553.

Ostermann S., Plankensteiner K., Prodan R., Fahringer T. (2011) GroudSim: An Event-Based Simulation Framework for Computational Grids and Clouds. Euro-Par 2010 Parallel Processing Workshops. Lecture Notes in Computer Science, vol 6586. Springer, Berlin, Heidelberg: 305-313. DOI: 10.1007/978-3-642-21878-1_38.

Pahl, C., Brogi, A., Soldani, J. 2017. Cloud Container Technologies: A State-of-the-Art Review. IEEE Transactions on Cloud Computing: 1 - 1. DOI: 10.1109/TCC.2017.2702586.

Pandit, D., Chattopadhyay, S., Chattopadhyay, M., Chaki, N. 2014. Resource allocation in cloud using simulated annealing. In 2014 Applications and Innovations in Mobile Computing (AIMoC), 21-27. DOI: 10.1109/AIMOC.2014.6785514.

Pedram, M., Hwang, I. 2010. Power and Performance Modeling in a Virtualized Server System. 39th International Conference on Parallel Processing Workshops: 520 - 526. DOI: 10.1109/ICPPW.2010.76.

Pereira, J., Da Silva, E. O., Batista, T., Delicato, F. C., Pires, P. F., Khan S. U. 2017. Cloud Adoption in Brazil. IT Professional 19(2): 50 - 56. DOI: 10.1109/MITP.2017.27.

R, Sudeepa & Guruprasad, H S. 2014. Resource Allocation in Cloud Computing. International Journal of Modern Communication Technologies and Research 2(4): 19-21.

Raju, R., Babukarthik, R.G., Chandramohan, D., Dhavachelvan, P., Vengattaraman, T. 2013. Minimizing the makespan using Hybrid algorithm for cloud computing. 3rd IEEE International Advance Computing Conference (IACC): 957-962. DOI: 10.1109/IAdCC.2013.6514356.

Rastogi, G., Sushil, R. 2015. Cloud computing implementation: Key issues and solutions. 2nd International Conference on Computing for Sustainable Global Development (INDIACom): 320-324.

Sahal, R., Khafagy, M. H., Omara, F. A. 2016. A Survey on SLA Management for Cloud Computing and Cloud-Hosted Big Data Analytic Applications. International Journal of Database Theory and Application 9(4): 107-118. DOI: 10.14257/ijdta.2016.9.4.10.

Samo, J. A., Ahmed, Z., Shaikh, A. 2017. Advocating isolation of resources among multi-tenants by containerization in IaaS cloud model. International Conference on Innovations in Electrical Engineering and Computational Technologies (ICIEECT): 1-17. DOI: 10.1109/ICIEECT.2017.7916567.

Sathya Sofia, A., GaneshKumar, P. 2018. Multi-objective task scheduling to minimize energy consumption and makespan of cloud computing using NSGA-II. Journal of network and systems management 26(2): 463-485. DOI:10.1007/s10922-017-9425-0.

Semnanian, A. A., Pham, J., Englert, B., Wu, X. 2011. Virtualization technology and its Impact on computer hardware architecture. Eigth international conference on information technology: New generations: 719-724. DOI: 10.1109/ITNG.2011.127.

Seung-Hwan, L., Bikash, S., Gunwoo, N. 2009. MDCSim: A multi-tier data center simulation, platform. IEEE international conference on cluster computing and workshops: 1 - 9. DOI: 10.1109/CLUSTR.2009.5289159.

Takako Endo, P., Santos Batista, M., Estácio Gonçalves, G. 2013. Self-organizing strategies for resource management in cloud computing: state-of-the-art and challenges. 2nd IEEE latin american conference on cloud computing and communications, 13-18. DOI: 10.1109/LatinCloud.2013.6842215.

Tchernykh, A., Schwiegelsohn, U., Yahyapour, R., Kuzjurin, N. 2010.Online hierarchical job scheduling on grids with admissible allocation. Journal of scheduling springer-verlag, Holanda: 13(5). 545-552. DOI: 10.1007/s10951-010-0169-x.

Tian, W., He, M., Guo, W., Huang, W., Shi, X., Shang, X., Toosi, A. N., Buyya, R. 2018. On minimizing total energy consumption in the scheduling of virtual machine reservations. Journal of network and computer applications 113: 64-74. DOI: 10.1016/j.jnca.2018.03.033.

Tikar, A. P., Jaybhaye, S. M., Pathak, G. R. 2015. A systematic review on scheduling types, methods and simulators in cloud computing system. International conference on applied and theoretical computing and communication technology (iCATccT): 382-388. DOI:10.1109/ICATCCT.2015.7456914.

Tsafrir, D., Etsion, Y., Feitelson, D. G. 2007.Backfilling using system-generated predictions rather than user runtime estimates. IEEE transactions on parallel and distributed systems 18(6): 789-803. DOI: 10.1109/TPDS.2007.70606.

Wu, L., Garg, S. K., Buyya, R.2011.SLA-based resource allocation for software as a service provider (SaaS) in cloud computing environments. 11th IEEE/ACM international symposium on cluster, cloud and grid computing: 195-204. DOI: 10.1109/CCGrid.2011.51.

Zhao, D., Mandagere,N., Alatorre,G., Mohamed,M., Ludwig, H.2015.Toward locality-aware scheduling for containerized cloud services. IEEE international conference on big data: 263-270. DOI: 10.1109/BigData.2015.7363763.

Zhao, D., Mohamed, M., Ludwig, H.2018.Locality-aware scheduling for containers in cloud computing. IEEE transactions on cloud computing: 1 - 14. DOI: 10.1109/TCC.2018.2794344.

Anexos

Anexo 1. Rangos y degradación media para estrategias en contenedores con modelo de capacidad requerida.

Estrategia	SLA	C_{max}	Energía	Media	Rango SLA	Rango C_{max}	Rango Energía	Rango Media
BFit_MaxCap_MinSla	0.287	0.167	0.028	0.161	14	36	14	15
BFit_MinCap_MinSla	0.267	0.093	0.019	0.126	11	6	1	6
BFit_MinSla_MinSla	0.201	0.197	0.039	0.146	6	48	35	11
BFit_MinTask_MinSla	0.175	0.152	0.028	0.118	5	27	12	5
BFit_MaxCap_MinTask	0.310	0.189	0.032	0.177	19	43	20	21
BFit_MinCap_MinTask	0.295	0.132	0.020	0.149	16	18	2	13
BFit_MinSla_MinTask	0.106	0.195	0.035	0.112	1	46	26	3
BFit_MinTask_MinTask	0.106	0.101	0.026	0.078	2	8	7	1
FFit_MaxCap_MinSla	0.527	0.130	0.030	0.229	38	17	17	34
FFit_MinCap_MinSla	0.514	0.090	0.025	0.210	37	5	5	30
FFit_MinSla_MinSla	0.397	0.225	0.050	0.224	25	54	49	33
FFit_MinTask_MinSla	0.400	0.142	0.036	0.193	27	21	28	23
FFit_MaxCap_MinTask	0.593	0.143	0.038	0.258	45	22	32	42
FFit_MinCap_MinTask	0.591	0.123	0.028	0.247	44	12	11	40
FFit_MinSla_MinTask	0.270	0.177	0.042	0.163	12	40	40	17
FFit_MinTask_MinTask	0.257	0.127	0.027	0.137	9	14	8	8
MaxSla_MaxCap_MinSla	0.499	0.203	0.036	0.246	35	50	27	38
MaxSla_MinCap_MinSla	0.490	0.072	0.024	0.195	33	1	3	24
MaxSla_MinSla_MinSla	0.408	0.195	0.048	0.217	28	45	46	32
MaxSla_MinTask_MinSla	0.397	0.161	0.034	0.197	24	33	23	27
MaxSla_MaxCap_MinTask	0.576	0.172	0.039	0.262	43	39	36	45
MaxSla_MinCap_MinTask	0.556	0.113	0.027	0.232	40	10	9	36
MaxSla_MinSla_MinTask	0.278	0.150	0.037	0.155	13	25	30	14
MaxSla_MinTask_MinTask	0.258	0.099	0.034	0.130	10	7	21	7
MinAW_MaxCap_MinSla	1.105	0.165	0.052	0.441	56	34	51	56
MinAW_MinCap_MinSla	0.771	0.077	0.037	0.295	49	2	29	47
MinAW_MinSla_MinSla	0.651	0.236	0.063	0.317	48	55	56	49
MinAW_MinTask_MinSla	0.639	0.186	0.063	0.296	47	42	55	48
MinAW_MaxCap_MinTask	0.949	0.196	0.056	0.400	55	47	53	55
MinAW_MinCap_MinTask	0.918	0.154	0.046	0.373	54	28	44	53
MinAW_MinSla_MinTask	0.502	0.203	0.056	0.254	36	49	54	41
MinAW_MinTask_MinTask	0.495	0.144	0.049	0.229	34	23	48	35
MinSla_MaxCap_MinSla	0.459	0.150	0.034	0.214	29	24	22	31
MinSla_MinCap_MinSla	0.479	0.087	0.024	0.197	32	4	4	26
MinSla_MinSla_MinSla	0.319	0.237	0.043	0.199	20	56	41	28
MinSla_MinTask_MinSla	0.298	0.151	0.038	0.162	18	26	31	16
MinSla_MaxCap_MinTask	0.544	0.157	0.039	0.247	39	30	34	39
MinSla_MinCap_MinTask	0.619	0.128	0.028	0.258	46	15	13	43
MinSla_MinSla_MinTask	0.295	0.180	0.043	0.173	17	41	42	19

MinSla_MinTask_MinTask	0.288	0.124	0.030	0.147	15	13	18	12
Rand	9.282	0.796	0.262	3.447	58	58	58	58
Rand_MaxCap_MinSla	0.334	0.155	0.035	0.175	22	29	25	20
Rand_MinCap_MinSla	0.325	0.081	0.026	0.144	21	3	6	10
Rand_MinSla_MinSla	0.231	0.223	0.049	0.167	8	53	47	18
Rand_MinTask_MinSla	0.218	0.169	0.035	0.141	7	37	24	9
Rand_MaxCap_MinTask	0.399	0.159	0.030	0.196	26	32	16	25
Rand_MinCap_MinTask	0.374	0.136	0.031	0.180	23	20	19	22
Rand_MinSla_MinTask	0.150	0.158	0.039	0.116	4	31	37	4
Rand_MinTask_MinTask	0.129	0.130	0.030	0.096	3	16	15	2
RR	8.575	0.686	0.241	3.167	57	57	57	57
WFit_MaxCap_MinSla	0.867	0.171	0.042	0.360	51	38	39	52
WFit_MinCap_MinSla	0.862	0.102	0.028	0.330	50	9	10	50
WFit_MinSla_MinSla	0.575	0.205	0.050	0.277	42	52	50	46
WFit_MinTask_MinSla	0.572	0.166	0.045	0.261	41	35	43	44
WFit_MaxCap_MinTask	0.909	0.203	0.047	0.386	53	51	45	54
WFit_MinCap_MinTask	0.894	0.136	0.039	0.356	52	19	33	51
WFit_MinSla_MinTask	0.463	0.190	0.052	0.235	31	44	52	37
WFit_MinTask_MinTask	0.461	0.118	0.041	0.207	30	11	38	29

Anexo 2. Rangos y degradación media para las estrategias de los dos escenarios estudiados.

Estrategia	SLA	Energía	C_{max}	Media	Rango SLA	Rango Energía	Rango C_{max}	Rango Media
BFit_MaxCap_MinSla	7.1276	0.0546	0.1856	2.4559	22	26	48	22
BFit_MaxCap_MinTask	7.0853	0.0448	0.1122	2.4141	21	8	11	21
BFit_MinCap_MinSla	6.8362	0.0652	0.2172	2.3729	20	45	61	20
BFit_MinCap_MinTask	6.7897	0.0542	0.1728	2.3389	19	24	39	19
BFit_MinSla	0.7552	0.0476	0.1482	0.3170	7	13	26	7
BFit_MinSla_MinSla	7.6707	0.0586	0.2080	2.6458	28	32	56	28
BFit_MinSla_MinTask	7.6507	0.0460	0.1509	2.6159	27	10	29	27
BFit_MinTask	0.7384	0.0247	0.0972	0.2867	5	1	6	5
BFit_MinTask_MinSla	5.1066	0.0616	0.2151	1.7944	14	38	58	16
BFit_MinTask_MinTask	5.1088	0.0525	0.1205	1.7606	16	17	14	14
FFit_MaxCap_MinSla	12.5785	0.0568	0.1489	4.2614	46	30	27	46
FFit_MaxCap_MinTask	12.5542	0.0520	0.1094	4.2385	45	16	9	45
FFit_MinCap_MinSla	12.1286	0.0769	0.2457	4.1504	43	62	68	44
FFit_MinCap_MinTask	12.1334	0.0627	0.1620	4.1194	44	40	33	43
FFit_MinSla	13.9390	0.0535	0.2249	4.7391	56	19	65	56
FFit_MinSla_MinSla	15.5440	0.0648	0.1630	5.2573	62	44	34	64
FFit_MinSla_MinTask	15.5450	0.0541	0.1415	5.2469	63	23	21	62
FFit_MinTask	1.2047	0.0418	0.1095	0.4520	9	4	10	9
FFit_MinTask_MinSla	10.5678	0.0690	0.1963	3.6110	36	53	53	37
FFit_MinTask_MinTask	5.1088	0.0535	0.1460	1.7694	15	20	23	15
MaxSla_MaxCap_MinSla	11.7724	0.0626	0.2221	4.0191	42	39	62	42
MaxSla_MaxCap_MinTask	11.7528	0.0503	0.0903	3.9645	41	15	2	39
MaxSla_MinCap_MinSla	13.1365	0.0754	0.2152	4.4757	52	59	59	52
MaxSla_MinCap_MinTask	13.1156	0.0605	0.1813	4.4525	51	34	46	51
MaxSla_MinSla	1.2595	0.0433	0.1568	0.4866	10	6	32	10
MaxSla_MinSla_MinSla	14.1438	0.0657	0.1910	4.8002	60	48	52	60
MaxSla_MinSla_MinTask	14.1077	0.0538	0.1322	4.7646	59	21	17	57
MaxSla_MinTask	0.6837	0.0350	0.1141	0.2776	3	2	12	3
MaxSla_MinTask_MinSla	10.3264	0.0629	0.1701	3.5198	34	41	37	34
MaxSla_MinTask_MinTask	10.5454	0.0602	0.1183	3.5746	35	33	13	35
MinAW_MaxCap_MinSla	20.5585	0.0791	0.1843	6.9406	70	64	47	70
MinAW_MaxCap_MinTask	13.4093	0.0639	0.0952	4.5228	53	42	4	53
MinAW_MinCap_MinSla	13.7360	0.0904	0.2569	4.6944	55	69	69	55
MinAW_MinCap_MinTask	13.7158	0.0905	0.2064	4.6709	54	70	55	54
MinAW_MinSla	0.8311	0.0470	0.1460	0.3414	8	12	24	8
MinAW_MinSla_MinSla	18.5566	0.0830	0.2154	6.2850	67	66	60	67
MinAW_MinSla_MinTask	18.4991	0.0729	0.1737	6.2485	66	57	40	66
MinAW_MinTask	0.2075	0.0426	0.0966	0.1156	2	5	5	2
MinAW_MinTask_MinSla	12.6509	0.0836	0.2229	4.3192	48	67	63	48

MinAW_MinTask_MinTask	12.6389	0.0761	0.1643	4.2931	47	61	35	47
MinSla_MaxCap_MinSla	9.4152	0.0607	0.1696	3.2151	30	35	36	30
MinSla_MaxCap_MinTask	10.6128	0.0502	0.1060	3.5897	37	14	8	36
MinSla_MinCap_MinSla	9.0635	0.0693	0.2572	3.1300	29	54	70	29
MinSla_MinCap_MinTask	10.0110	0.0644	0.1715	3.4156	33	43	38	33
MinSla_MinSla	1.4615	0.0459	0.1356	0.5477	11	9	18	11
MinSla_MinSla_MinSla	11.6910	0.0657	0.1767	3.9778	40	47	42	41
MinSla_MinSla_MinTask	14.3484	0.0545	0.1473	4.8501	61	25	25	61
MinSla_MinTask	0.6849	0.0369	0.1207	0.2808	4	3	15	4
MinSla_MinTask_MinSla	11.6502	0.0701	0.1993	3.9732	39	55	54	40
MinSla_MinTask_MinTask	11.6379	0.0567	0.1435	3.9461	38	29	22	38
Rand	129.4753	0.2953	0.8270	43.5325	72	72	72	72
Rand_MaxCap_MinSla	7.5322	0.0614	0.1747	2.5894	26	37	41	26
Rand_MaxCap_MinTask	7.5137	0.0527	0.0996	2.5553	25	18	7	25
Rand_MinCap_MinSla	7.2961	0.0757	0.2430	2.5383	24	60	67	24
Rand_MinCap_MinTask	7.2713	0.0610	0.1890	2.5071	23	36	50	23
Rand_MinSla	2.6977	0.0864	0.1795	0.9878	13	68	45	13
Rand_MinSla_MinSla	9.8005	0.0564	0.1787	3.3452	32	27	44	32
Rand_MinSla_MinTask	9.7534	0.0583	0.1554	3.3224	31	31	30	31
Rand_MinTask	2.6324	0.0677	0.0943	0.9315	12	50	3	12
Rand_MinTask_MinSla	6.6794	0.0661	0.1772	2.3076	18	49	43	18
Rand_MinTask_MinTask	6.6419	0.0565	0.1497	2.2827	17	28	28	17
RR	122.2884	0.2744	0.7154	41.0927	71	71	71	71
WFit_MinSla	0.7543	0.0469	0.1404	0.3139	6	11	20	6
WFit_MinTask	0.1853	0.0444	0.0809	0.1035	1	7	1	1
WFit_MaxCap_MinSla	15.5838	0.0684	0.1906	5.2810	65	52	51	65
WFit_MaxCap_MinTask	15.5708	0.0540	0.1218	5.2489	64	22	16	63
WFit_MinCap_MinSla	14.0517	0.0776	0.2264	4.7852	58	63	66	59
WFit_MinCap_MinTask	14.0418	0.0723	0.1872	4.7671	57	56	49	58
WFit_MinSla_MinSla	19.4125	0.0738	0.2236	6.5700	69	58	64	69
WFit_MinSla_MinTask	19.3814	0.0654	0.1555	6.5341	68	46	31	68
WFit_MinTask_MinSla	12.9113	0.0794	0.2101	4.4003	50	65	57	50
WFit_MinTask_MinTask	12.9065	0.0681	0.1378	4.3708	49	51	19	49