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

Doctorado en Ciencias

en ciencias de la computación

Modelo de energía basado en la concentración de tareas heterogéneas y diseño de estrategias de consolidación en cómputo en la nube

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

Presenta:

Fermin Alberto Armenta Cano

Ensenada, Baja California, México
2018

Tesis defendida por

Fermin Alberto Armenta Cano

y aprobada por el siguiente Comité

Dr. Andrey Chernykh
Director de tesis

Miembros del comité
Dr. Carlos Alberto Brizuela Rodríguez

Dr. Edgar Leonel Chávez González

Dr. Adan Hirales Carbajal

Dr. Ramin Yahyapour

Dr. Jesús Favela Vara
Coordinador del Posgrado en Ciencias computacionales

Dra. Rufina Hernández Martínez
Directora de Estudios de Posgrado

Resumen de la tesis que presenta Fermin Alberto Armenta Cano como requisito parcial para la obtención del grado de Doctor en Ciencias en Doctor en Ciencias en Ciencias de la Computación

Modelo de energía basado en la concentración de tareas heterogéneas y diseño de estrategias de consolidación en cómputo en la nube

Resumen aprobado por:

_____

Dr. Andrey Chernykh
Director de tesis

La ejecución de tareas que comparten recursos de hardware en un mismo servidor, puede crear contención en el uso compartido de CPU, memoria, disco o red. Lo cual conduce a la degradación del desempeño del sistema e incrementan el consumo energético. En esta tesis, presentamos un modelo de energía novedoso analizando el consumo de energía de un servidor al ejecutar diferentes clases de aplicaciones. Definimos el modelo de energía en un escenario realístico al ejecutar cargas de trabajo heterogéneas. Consideramos aplicaciones intensivas en CPU y memoria. El modelo de energía propuesto considera el consumo energético durante la ejecución de cada tipo de tarea, así como el consumo energético al ejecutar combinaciones entre los diferentes tipos de tareas. Cada tipo de tarea requiere diferentes partes del hardware contribuyendo de forma diferente al consumo de energía total del sistema. En este trabajo, se considera el consumo de energía en función de la utilización del CPU y la concentración de diferentes clases de tareas. Analizamos múltiples algoritmos de planificación de tareas los cuales requieren diferente tipo y cantidad de información. Diseñamos estrategias de calendarización novedosas que consideran el agrupamiento de trabajos heterogéneos. Validamos la calidad de cada estrategia realizando un análisis detallado utilizando cargas de trabajo reales en diferentes escenarios con ayuda del simulador Cloudsim. El modelo de estimación de energía propuesto se basa en mediciones del consumo de energía a nivel del procesador con la herramienta likwid-powermeter y a nivel del servidor con la unidad especializada de distribución de energía (PDU). Finalmente, en este trabajo se demuestra que las estrategias propuestas mejoran la eficiencia energética con respecto a estrategias clásicas de la literatura. Las principales aportaciones de esta tesis son las siguientes: Construimos el perfil energético de diferentes tipos de trabajo y sus combinaciones. Detectamos una reducción en el consumo de energía del servidor cuando se considera el equilibrio de concentración de aplicaciones en el procesador. Identificamos cómo la degradación del rendimiento afecta el consumo de energía cuando se produce contención de recursos. Diseñamos estrategias que asignan trabajos en servidores para minimizar la contención de recursos. Finalmente, evaluamos el rendimiento de 10 estrategias de asignación de recursos por medio de experimentación.

Palabras clave: Eficiencia energética, Modelo de energía, Tipos de aplicaciones, Contención de recursos, Medición de energía, Planificación de tareas.

Abstract of the thesis presented by Fermin Alberto Armenta Cano as a partial requirement to obtain the Doctor of Science degree in Computer Sciences

Heterogeneous Jobs Concentration Energy Model and Consolidation Strategies in Cloud Computing

Abstract approved by:

_____

Dr. Andrey Chernykh
Thesis Director

An execution of similar jobs in shared infrastructure could create a resource contention either in CPU, memory, disk and network. It causes degradation of the system performance and increased energy consumption. In this paper, we analyze a server energy consumption, when it executes CPU-intensive and memory-intensive applications. We define a power consumption model for realistic scenarios of heterogeneous workload. Our model takes into account power consumption of individual jobs and their combinations. Different types of applications exploit different hardware and contribute differently to the total power consumption. We consider power consumption as a function of CPU utilization and job concentration. We analyze several scheduling algorithms depending on the type and amount of information they require. We propose novel heterogeneous job consolidation policies and validate them by a comprehensive evaluation analysis on real data in different scenarios using CloudSim toolkit. The energy consumption is measured at processor level with likwid-powermeter and at the server level with a specialized power distribution unit. We demonstrate that our solutions outperform known state-of-the-art strategies. The key contributions of this thesis are the following. We construct the energy profile of different job types and their combinations. We propose an energy model that consider the job concentration. We detect a reduction in a server energy consumption when the balance of concentration of applications in the same processor is considered. We identify how performance degradation affects the energy consumption when contention occurs. We design strategies that allocate jobs on servers to minimize resource contention and finally, we evaluate the performance of 10 resource allocation strategies using experimentation.

Keywords: Energy Efficiency, Energy model, Type of applications, Contention, Energy measurement, Scheduling.

Dedicatoria

A mi hija Allison, que me ha enseñado a comprender que los pequeños detalles de la vida son los realmente importantes.

A mis padres Don Coruco y Ñoda Tere, muchas gracias por todo el apoyo y amor incondicional que siempre me han brindado.

A Vanessa, muchas gracias por motivarme a continuar en este camino que algunos llaman investigacion y por soportar con pena ajena todas mis imprudencias.

A mis amigos e investigadores de ciencias de la computación, en especial a las “Maniwis” y a ese grupo selecto con el que compartí exquisitos desayunos a muy tempranas horas.

Agradecimientos

Me gustaría expresar mi profunda gratitud y mi más sincero agradecimiento a mi asesor, el Profesor Dr. Andrey Chernykh, por brindarme el apoyo y el estímulo para llevar a cabo mi investigación en los últimos cuatro años.

Mi profundo agradecimiento a CICESE y a todos los miembros de mi comité de tesis, los profesores Dr. Carlos Brizuela, Dr. Edgar Chávez, Dr. Adan Hirales y el Dr. Ramin Yahyapour por los consejos y el interés en ayudarme en la elaboracion de esta tesis.

Mi agradecimiento a todos mis colegas coautores por su participación en las publicaciones derivadas de mi tesis doctoral. Las cuales ayudaron en la elaboración de este documento.

Recuerdo con gratitud la ayuda desinteresada y la participación de mis compañeros de laboratorio Vanessa Miranda, Julia Díaz, Mario Cortés, Rewer Canosa, Luis Galaviz y Manuel Combarro.

Finalmente, agradezco a todas las personas que hacen posible la investigación en México. Gracias a todo el pueblo de México que por medio del Consejo Nacional de Ciencia y Tecnología (CONACyT) me otorgaron una beca (número de becario 225073) para la realización de este trabajo de investigación.

Tabla de Contenido

Lista de figuras

Figura **Página**

Lista de tablas

Tabla **Página**

Capítulo 1. Introducción

La computación en la nube es un modelo distribuido que opera bajo demanda, y es ampliamente aceptado por organizaciones públicas y privadas. El Instituto Nacional de Estándares y Tecnología (NIST, por sus siglas en inglés) define a la computación 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, los cuales pueden ser configurados bajo demanda. Por ejemplo : redes, servidores, almacenamiento, aplicaciones y servicios. Este servicio puede ser rápidamente abastecido y activado con un mínimo esfuerzo de gestión o interacción del proveedor del servicio ". NIST define las siguientes características esenciales de la nube (Mell y Grance, 2011):

- Autoservicio: los consumidores pueden acceder bajo demanda a una serie de recursos informáticos compartidos, tales como el procesamiento y el almacenamiento del servidor, según sea necesario, sin interacción humana con cada proveedor de la nube.
- Recursos compartidos: los recursos informáticos se agrupan para servir a varios usuarios mediante un modelo de organización de múltiples clientes. Varios recursos físicos son asignados dinámicamente y reasignados de acuerdo con la demanda del consumidor. Hay una independencia de ubicación dado que el cliente no tiene información sobre la ubicación física de los recursos proporcionados. Sin embargo, el cliente puede especificar una ubicación como país, estado o centro de datos. Los ejemplos de recursos incluyen almacenamiento, procesamiento, memoria y ancho de banda de red.
- Amplio acceso a la red: los clientes pueden acceder a través de mecanismos estándar utilizados por plataformas heterogéneas de diferentes clientes a través de la red (por ejemplo, teléfonos móviles, tabletas, computadoras portátiles y estaciones de trabajo).
- Elasticidad: los recursos informáticos pueden escalarse automáticamente y adecuarse a la demanda. Las habilidades disponibles para abastecer al consumidor a menudo parecen ser ilimitadas y pueden asignarse en cualquier cantidad y en cualquier momento.
- Servicio medido: las organizaciones de la nube supervisan y mejoran automáticamente la utilización de los recursos, implementando un sistema de medición adecuado para cada servicio (por ejemplo, procesamiento, almacenamiento y ancho de banda). La utilización de los recursos puede ser monitoreada, controlada e informada, proporcionando transparencia tanto para el proveedor como para el consumidor del servicio utilizado.

Los proveedores de la nube ofrecen recursos computacionales y servicios que garantizan los requisitos de calidad de servicio (QoS, por sus siglas en inglés). Los costos de operación de la nube están relacionados principalmente con la infraestructura con la que brinda el servicio y el mantenimiento de la misma. La infraestructura con la que operan los proveedores de la nube debe ser energizada y operar a una temperatura adecuada.

El gasto de energía es uno de los principales componentes del costo operativo y, por lo tanto, una de las principales preocupaciones del proveedor. Por lo tanto, la eficiencia energética, es esencial para los proveedores de nubes. Existen diferentes enfoques para mejorar la eficiencia energética. Algunos de ellos son: optimizar la tecnología de hardware, escala dinámica de voltaje y frecuencia, por sus siglas en inglés (DVFS), optimización de código de software y gestión de recursos, entre otros.

Los centros de datos deben reducir el rastro de carbono y compensar los aumentos notables en los gastos de energía. Por lo tanto, las medidas de ahorro de energía a nivel tecnológico son obligatorias para cualquier tecnología emergente de información y comunicación. En este contexto, los proveedores de la nube deben evaluar diferentes estrategias para mejorar la eficiencia energética, incluidos los equipos de cómputo, refrigeración y suministro de energía. Esta evaluación implica la definición y el uso de métricas unificadas, como PUE (Power Usage Effectiveness) la cual mide con qué eficiencia consume energía un centro de datos, o la métrica DCIE (Data Center Infrastructure Efficiency) la cual determina la eficiencia energética de un centro de datos, entre otras. Estas métricas ayudan a los proveedores de la nube a medir la eficiencia energética del servidor, compararlo con otras infraestructuras de la nube y decidir qué mejoras son necesarias.

La optimización de la eficiencia energética continúa en aumento, como se refleja en los informes del Green500 del mes de junio de 2017. Durante 2017, las supercomputadoras más ecológicas del mundo duplicaron su eficiencia energética. Es el salto más significativo desde que el Green500 comenzó a clasificar estos sistemas hace más de una década. La Figura 1 muestra la evolución de las supercomputadoras mejor clasificadas en Green500 durante la última década. Desde junio de 2016 hasta junio de 2017, el promedio de eficiencia energética de los diez principales sistemas saltó de 4.8 Gflops por watt a 11.1 Gflops por watt, lo que representa un aumento de 2.3x en 12 meses. Esa mejora fue principalmente el resultado del despliegue de supercomputadoras equipadas con NVIDIA Tesla P100 GPU, que están presentes en nueve de las diez principales máquinas (Michael Feldman, 2017).

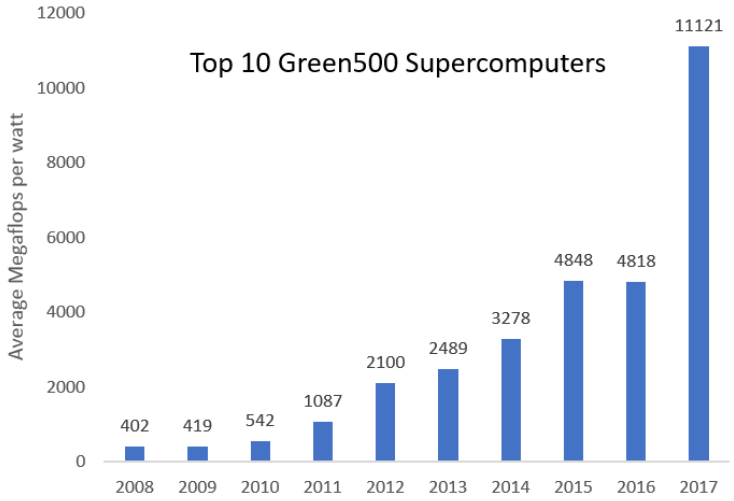


Figura 1. Progresión de las supercomputadoras mejor clasificadas en el Green500 (Michael Feldman, 2017)

En entornos compartidos, a menudo es difícil optimizar el consumo de energía de los recursos físicos con diferentes tipos de trabajos. Los cuatro principales tipos de trabajo son: intensivo en CPU (CI), intensivo en disco (DI), intensivo en memoria (MI) e intensivo en red (NI). Esta tesis se centra en el manejo eficiente de recursos para lograr la optimización del consumo de energía.

Mediciones exactas del consumo de energía en un servidor, son esenciales en el proceso de elaboración de un modelo de energía. Dicho modelo es parte fundamental en el diseño de estrategias que ayuden a optimizar energía. En esta tesis, para obtener mediciones precisas y exactas del consumo de energía en un servidor cuando se ejecutan una o más aplicaciones, utilizamos una unidad de distribución de energía (PDU). Por otro lado, para obtener las mediciones de consumo energético a nivel del procesador, utilizamos likwid-powermeter, una herramienta de software que permite el acceso a los contadores de programa RAPL (por sus siglas en inglés, Running Average Power Limit) en los procesadores Intel.

Dado que el modelo de energía debe brindar información exacta a las políticas de asignación conscientes de la energía. Llevamos a cabo un conjunto de evaluaciones empíricas para determinar las ecuaciones apropiadas para el modelo de energía propuesto.

1.1 Cómputo en la nube

En este capítulo, presentamos conceptos esenciales de cómputo en la nube relacionados con nuestra investigación. Definimos los niveles de servicio como una herramienta que establece las pautas sobre las características y la calidad del servicio que reciben los usuarios finales.

El Cómputo en la nube se centra en brindar servicios a sus clientes a través de internet con cierto nivel de calidad de servicio. Los tres principales servicios que se brindan son: Software como servicio (SaaS), Plataforma como servicio (PaaS) e Infraestructura como servicio (IaaS). Para especificar las obligaciones mínimas del proveedor hacia los clientes, se cuenta con una política de Acuerdos de nivel de servicio (SLA, por sus siglas en inglés). Las políticas de SLA son obligatorias debido a la naturaleza dinámica de la nube.

En esta tesis consideramos un modelo en un entorno basado en la infraestructura como servicio (IaaS), en donde se ofrecen recursos informáticos como CPU y memoria, bajo ciertas condiciones que aseguren la calidad de servicio.

1.1.1 Modelos de servicio

En esta subsección, presentamos los tres principales modelos de servicios que se ofrecen en cómputo en la nube.

Software como servicio (SaaS), es un modelo mediante el cual un proveedor ofrece una aplicación de software a través de internet, en lugar de un paquete de software para que el cliente lo compre. Algunos ejemplos son los proveedores de correo electrónico en línea como Gmail de Google, Microsoft Hotmail, asi como, Google Docs y Microsoft Office 365. El proveedor generalmente aloja y administra una aplicación determinada en su centro de datos y la pone a disposición para múltiples usuario.

Plataforma como servicio (PaaS), proporciona una plataforma para ejecutar las aplicaciones del usuario final sin descargas ni instalación. Un ejemplo es Google App Engine, que permite ejecutar aplicaciones en la infraestructura de Google. Este servicio es una plataforma de desarrollo que admite todo el "Ciclo de vida del software", que permite a los consumidores de la nube desarrollar aplicaciones directamente en la nube PaaS. La principal diferencia entre SaaS y PaaS es que SaaS aloja las aplicaciones finalizadas en la nube, mientras que PaaS ofrece una plataforma de desarrollo que aloja tanto las aplicaciones en la nube en progreso como las finalizadas. SaaS requiere que PaaS, además de admitir un entorno de alojamiento

de aplicaciones, posea una infraestructura de desarrollo que incluya entornos de programación, herramientas, gestión de configuración, etc.

Infraestructura como servicio (IaaS), los proveedores ofrecen recursos de hardware como: capacidad de almacenamiento, capacidad de procesamiento y de memoria. En este modelo, los clientes sólo alquilan los recursos indispensables para sus necesidades en lugar de comprar todo el equipo. Uno de los principales proveedores es Amazon Web Services (EC2 y S3) para procesamiento y almacenamiento. La virtualización se usa ampliamente en la nube para aumentar la utilización de los recursos físicos con el objetivo de utilizar los recursos requeridos por los clientes de manera eficiente.

Los tres modelos de servicios en la nube requieren del acuerdo de nivel de servicio (SLA), que es un componente de extrema importancia en cómputo en la nube. El SLA representa un contrato que especifica las obligaciones mínimas del proveedor para sus clientes (las expectativas de los clientes a recibir a cambio del precio pagado). Los atributos de QoS que forman parte de un SLA (como el tiempo de respuesta y el rendimiento) deben supervisarse cuidadosamente para detectar violaciones al SLA, las cuales deben de ser sancionadas (Fortino, 2013).

1.1.2 Modelo de implementación

Actualmente existen tres modelos principales de implementación de sistemas de cómputo en la nube: nube privada, nube pública y nube híbrida. Cada uno de estos sistemas en la nube fue diseñado para cubrir las diversas necesidades de los clientes. La nube se puede compartir entre un número limitado de socios de confianza o ser servicios accesibles públicamente. Los modelos de implementación presentan algunas ventajas y desventajas sobre cómo los clientes pueden controlar, ampliar y acceder a los recursos. A continuación, presentamos una breve descripción de los modelos de implementación esenciales de los sistemas de computación en la nube (Chuvakin et al., 2013).

Nube privada: la infraestructura de la nube es mantenida exclusivamente por una organización. Puede ser administrada por la misma corporación o un tercero y puede existir dentro o fuera de las instalaciones de la organización.

Nube pública: la infraestructura en la nube se ofrece al público en general o a grupos industriales y es mantenida por una organización que vende servicios en la nube.

Nube híbrida: la infraestructura en la nube es una estructura típica de nubes privadas y públicas que siguen siendo entidades privadas pero están unidas por tecnología estandarizada o patentada que permite la portabilidad de los datos y las aplicaciones.

1.2 Heterogeneidad en la asignación de la carga de trabajo

Como se ha establecido, una de las principales preocupaciones del proveedor es el costo operacional, y por lo tanto, el gasto de energía. Algunos temas de interés que pueden ayudar a minimizar los costos de operación es mejorar el proceso de asignación de recursos en la nube, la consolidación y las estrategias de migración que consideran el perfil de las cargas de trabajo. En los entornos compartidos, a menudo es difícil optimizar el consumo de energía de los recursos físicos y las máquinas virtuales (MV) al ejecutar diferente tipos de tareas (CI, MI, DI e NI).

La reasignación de aplicaciones con el fin de consolidación, puede ser eficiente desde una perspectiva de ahorro de energía, pero puede ser contraproducente para el rendimiento del servicio cuando las cargas de trabajo tienen requisitos de comunicaciones estrechamente vinculados (Combarro et al., 2016).

En un entorno virtual, la consolidación de la carga de trabajo se realiza para aumentar la utilización de los servidores. En este escenario, los diferentes niveles que conforman una aplicación, por ejemplo: el nivel de la base de datos, el módulo de acceso y, el nivel de gestión, se asignan juntos y dan lugar a diferentes tipos de carga de trabajo mayormente intensivas en CPU, E/S, red y memoria. La contención en los recursos compartidos puede dar lugar a la degradación del rendimiento de las aplicaciones, denominada "Interferencia". Es necesario analizar la degradación del rendimiento en un servidor, que se produce cuando se realiza el proceso de consolidación. Así, evitar la contención mientras intentamos asignar una carga de trabajo heterogénea y consciente de la energía. Las variaciones en el rendimiento pueden observarse no sólo por el impacto de la aplicación co-runner (es decir, una aplicación que corre simultáneamente) sino también por la heterogeneidad del hardware. Se puede lograr un mayor ahorro de energía equilibrando los tipos de trabajo para maximizar el uso de los recursos del servidor sin causar la degradación del rendimiento de las aplicaciones.

1.3 Planteamiento del problema

El ahorro de energía se ha convertido en un tema crucial, se estima que el consumo de energía de los centros de datos aumentará alrededor de 140 mil millones de kWh anuales para el año 2020. Esta cantidad de energía es igual a la producción anual de 50 plantas de energía y produce casi 150 millones de toneladas de CO2 anualmente (Delforge y Whitney, 2014). Por ello, la eficiencia energética es una de las principales preocupaciones de computación en la nube. El uso ineficiente de los recursos de cómputo tiene un efecto negativo en el rendimiento del sistema y el consumo de energía. Graham et al., (1979) demostraron que el problema de asignar trabajos en un conjunto de recursos heterogéneos es NP-completo.

El problema que se aborda en este trabajo de investigación se divide en dos partes: Dado un conjunto de recursos computacionales y un conjunto de trabajos heterogéneos. Considerando la utilización y la concentración de cada tipo de trabajo en el procesador. 1) Caracterizar el consumo de energía a nivel de procesador y de servidor, 2) Implementar políticas inteligentes de asignación de trabajos con el objetivo de mejorar la eficiencia energética.

1.4 Justificación

La reducción del consumo de energía se ha convertido en uno de los principales problemas de investigación tanto en el sector privado como en el académico. Esto se debe al alto impacto ambiental, económico y en el rendimiento de operación de los equipos de cómputo. Según el informe de Kaplan et al. (2008), un centro de datos consume tanta energía como 25,000 hogares. Además, los costos de energía de alimentar un centro de datos se duplican cada cinco años.

El monitoreo y manejo detallado del consumo energético debe ser tomado en cuenta para optimizar el uso de los recursos y mejorar la eficiencia energética (Kliazovich et al., 2016). En ambientes compartidos, los usuarios envían cargas de trabajo heterogéneas que incluyen el uso intensivo de CPU, I/O de disco, uso intensivo de memoria, intensivo de E/S de red entre otras aplicaciones. Cuando los trabajos de un tipo se asignan dentro del mismo recurso físico, puede crear conflictos de acceso al hardware tanto de CPU, memoria, disco o la red. La contención provoca la degradación del rendimiento del sistema y el aumento del consumo de energía.

1.5 Hipótesis

El consumo de energía del servidor no sólo depende de su utilización sino también del tipo de trabajos que está ejecutando. La eficiencia energética podría maximizarse si se considera el tipo de trabajo en el proceso de calendarización. El consumo de energía cuando se combinan diferentes tipos de aplicaciones en el mismo servidor, es menor que la suma de los consumos de energía de cada tipo de aplicación por separado.

Diseñar e implementar un modelo de energía teniendo en cuenta los tipos de aplicaciones, ayudará en el proceso de toma de decisiones de los algoritmos de asignación de recursos con el objetivo de minimizar el consumo total de energía.

1.6 Objetivos

1.6.1 Objetivo general

El objetivo general se compone de dos partes:
1) Diseñar, implementar y evaluar un modelo de energía basado en los perfiles energéticos de diferentes aplicaciones y sus combinaciones;
2) Diseñar, implementar y evaluar estrategias de asignación para minimizar el consumo total de energía en la ejecución de cargas de trabajo heterogénea. Tomando en cuenta los tipos de trabajo, al mismo tiempo que se garantizan los requisitos de calidad de servicio.

1.6.2. Objetivos específicos

- Obtener el perfil de consumo de energía de tareas intensivas en CPU (CI).
- Obtener el perfil de consumo de energía de tareas intensivas en memoria (MI).
- Obtener el perfil de consumo de energía de tareas intensivas en disco (DI).
- Obtener una función que calcula la corrección en la estimación de energía que se genera al combinar diferentes tipos de trabajo que se ejecutan en un servidor.
- Diseñar e implementar un modelo de energía basado en los perfiles energéticos de cada aplicación y la función de corrección.
- Adaptar el simulador CloudSim a un entorno en línea, que considere cargas de trabajo heterogéneas.

- Diseñar e implementar estrategias de calendarización consciente de la energía, considerando cargas de trabajo heterogéneas.
- Evaluar el rendimiento de las estrategias de calendarización propuestas por medio de la experimentación.

1.6.4 Contribución

Las principales contribuciones de este trabajo de investigación se enlistan a continuación:

- Analizamos y construimos el perfil energético de diferentes tipos de aplicaciones y sus combinaciones.
- Proponemos un modelo de energía que considera el consumo de energía como una función de la utilización del CPU y la concentración de los trabajos.
- Detectamos una reducción en el consumo de energía del servidor cuando se considera el equilibrio de la concentración de aplicaciones.
- Identificamos cómo la degradación en el rendimiento del sistema afecta el consumo de energía cuando se produce contención de recursos.
- Mostramos que al considerar la utilización del CPU y la concentración de aplicaciones ayuda en el proceso de toma de decisiones, cuando las aplicaciones se asignan al servidor.
- Diseñamos estrategias que asignan la carga de trabajo en los servidores para evitar la contención de recursos.
- Evaluamos el desempeño de 10 estrategias de asignación de recursos por medio de experimentación.

1.6.5 Organización de la tesis

Esta tesis está organizada de la siguiente manera. En el Capítulo 1, presentamos una introducción a la computación en la nube, el enunciado del problema, la justificación y los objetivos de la tesis. En el Capítulo 2, discutimos una serie de trabajos relacionados que implementan algoritmos de asignación de recursos conscientes de la energía, consciente de la contención y por último trabajos relacionado en el monitoreo de energía en un centro de datos. En el Capítulo 3, describimos el modelo de energía junto con la metodología de medición de energía. Además, presentamos un análisis de la energía consumida por diferentes aplicaciones para obtener los perfiles energéticos por tipo de aplicación. En el Capítulo 4,

presentamos los algoritmos de calendarización utilizados en nuestro estudio experimental, configuración experimental, una breve descripción del simulador CloudSim y la metodología del análisis. En el Capítulo 5, presentamos la validación experimental. En el Capítulo 6, presentamos las conclusiones de este trabajo y finalmente, el apartado de apéndice.

Capítulo 2. Antecedentes

2.1 Algoritmos de asignación de recursos conscientes de la energía

En la literatura, las estrategias más comunes de ahorro de energía en centros de datos son: Optimización de hardware, escala dinámica de voltaje y frecuencia (DVFS) y desactivación parcial o completa de componentes dinámicos (PDCD). En la técnica DVFS, la idea principal es supervisar la utilización del CPU y ajustar continuamente la frecuencia de reloj y el voltaje de alimentación para que coincidan con los requisitos de rendimiento que se demandan. Por otro lado, la técnica de PDCD asigna la carga de trabajo al mínimo número de servidores físicos que se requieren y desactiva los servidores inactivos.

A continuación, presentamos trabajos relacionados a nuestro tema de investigación, con un enfoque de manejo de los recursos que utiliza estrategias conscientes de la energía y diferentes modelos de consumo de energía.

EMVM- heurística de asignación de recursos consciente de la energía para una gestión eficiente, por Beloglazov et al. (2012). Los autores presentan algoritmos de asignación de recursos que utilizan la consolidación dinámica de máquinas virtuales y describen principios de gestión eficiente de la energía en entornos de computación en la nube. En su trabajo muestran que la consolidación conduce a una reducción sustancial del consumo de energía en comparación con las técnicas estáticas de asignación de recursos.

El modelo de consumo de energía que utilizan es:

$$P(u) = k * Pmax + (1 - k)Pmax * u$$

Donde $Pmax$ es el consumo de energía máximo cuando el servidor se utiliza por completo; k es la fracción de potencia consumida por el servidor inactivo (para este caso, 0.7); y u es la utilización del CPU.

El consumo total de energía E se define como una integral de la potencia consumida en un intervalo de tiempo dado:

$$E = \int_{t_0}^{t_1} P\big(u(t)\big)dt.$$

Cuando las máquinas virtuales no usan todos los recursos del procesador, se pueden consolidar en la mínima cantidad de procesadores físicos. Los procesadores inactivos se pueden cambiar a modo de suspensión, para minimizar el consumo de energía.

La Figura 2 muestra el consumo de energía de acuerdo con el modelo descrito variando la utilización del CPU.

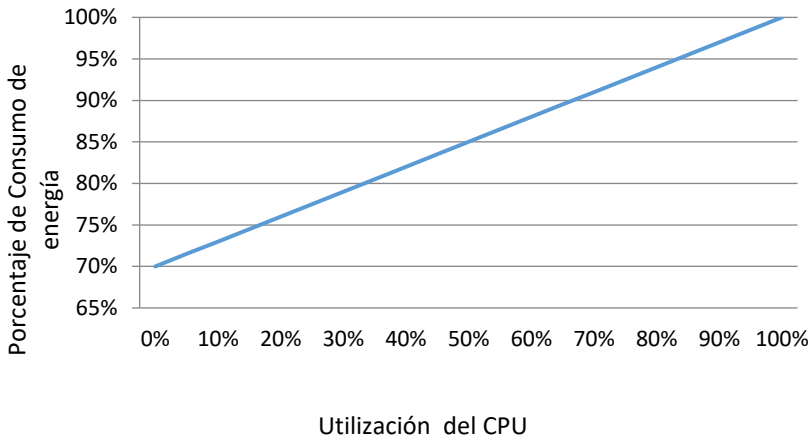


Figura 2. Consumo de energía lineal frente a la utilización del CPU (%).

HSFL- Algoritmo de salto de rana híbrido por Luo et al. (2014). El esquema de gestión de recursos utilizado en este trabajo, garantiza la calidad de servicio del usuario (QoS) especificada por el SLA, logrando un ahorro de energía. Los autores usan la técnica de migración de VMs para consolidar los recursos. Los servidores con baja utilización o inactivos se cambian al modo de ahorro de energía, lo que aumenta la eficiencia energética y garantiza la QoS.

El estudio supone un consumo de energía lineal del CPU. El consumo de energía de un CPU inactivo representa el 70% del consumo máximo de energía. El consumo de energía en un momento dado h se define de la siguiente manera:

$$E(h) = 0.7E_{max}(h) + 0.3Utlz(h)E_{max}(h) + 0.1E_{max}\sum_{i\in v} T(i).$$

Donde $E\_max\ (h)$ es el consumo de energía cuando el Servidor está operando a su máxima carga. $Utlz(h)$ es la tasa de utilización promedio del procesador, v es el conjunto de VMs migradas, y $T(i)$ es el tiempo

de migración de la i-ésima máquina virtual. La relación entre la utilización del CPU y el consumo de energía es similar a la presentada en la Figura 2.

AETC- Algoritmo de consolidación de tareas consciente de la energía, por Hsu et al. (2014). Los autores proponen una técnica de consolidación de tareas consciente de la energía (ETC), con el objetivo de minimizar el consumo energético. El algoritmo ETC está diseñado para funcionar en un centro de datos con máquinas virtuales que están siendo ejecutadas en máquinas físicas ubicadas en el mismo estante (rack) o estantes con un ancho de banda permanente. Además, restringe el uso del CPU por encima de un umbral máximo especificado del 70% mediante la consolidación de tareas entre los *clústeres* virtuales. Asimismo, para el costo de la energía considera la latencia de la red cuando un trabajo migra a otro *clúster* virtual. Se supone que el estado inactivo de las máquinas virtuales y la transmisión de red tienen una parte constante del consumo total de energía.

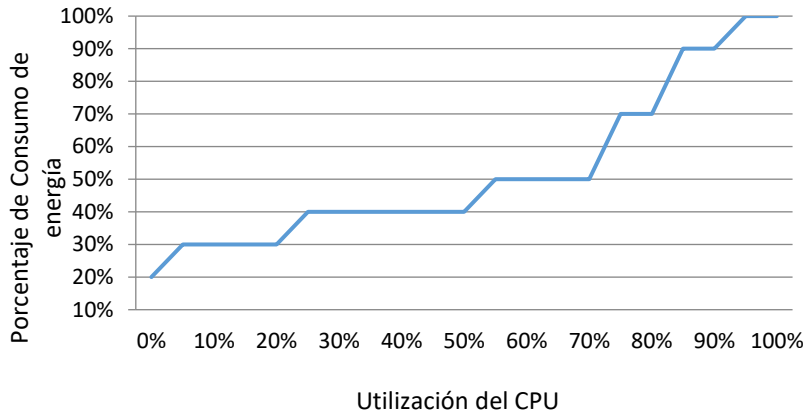


Figura 3. Consumo de energía escalonado frente a la utilización del CPU (%).

Los resultados presentados por Hsu et al. (2014) muestran que el algoritmo ETC puede reducir el consumo de energía al realizar una consolidación de tareas en los sistemas de la nube. La Figura 3 muestra el porcentaje de consumo de energía con respecto a la utilización del CPU que utiliza el algoritmo ETC.

El modelo supone que el consumo de energía en estado ocioso es $E(Vi) = \alpha\ W/s$. La energía adicional necesaria para ejecutar tareas cuando se incrementa la utilización del CPU se encuentra en función de β.

$$E_t(Vi) = \begin{cases} \alpha\, W/s, & si\ es\ ocioso \\ \beta + \alpha\, W/s, & si\ 0\% < CPU\ util \leq 20\% \\ 3\beta + \alpha\, W/s, & si\ 20\% < CPU\ util \leq 50\% \\ 5\beta + \alpha\, W/s, & si\ 50\% < CPU\ util \leq 70\% \\ 8\beta + \alpha\, W/s, & si\ 70\% < CPU\ util \leq 80\% \\ 11\beta + \alpha\, W/s, & si\ 80\% < CPU\ util \leq 90\% \\ 12\beta + \alpha\, W/s, & si\ 90\% < CPU\ util \leq 100\% \end{cases}$$

El consumo de energía de la máquina virtual $\mathrm{V_i}$ durante el periodo $[\mathrm{t_0}, \mathrm{t_m}]$ se define a continuación:

$$E_{0,m}(V_i) = \sum_{t=0}^{m} E_t(V_i).$$

Para un clúster virtual dado, $\mathrm{VC_k}$ el cual consiste de n VMs, el consumo de energía se calcula de la siguiente forma:

$$E_{0,m}(VC_k) = \sum_{i=0}^{n} E_{0,m}(V_i).$$

CTES- Calendarización cooperativa de dos niveles consciente de la energía, por Hosseinimotlagh et al. (2014). Los autores abordan un enfoque cooperativo de calendarizacion de dos niveles de tareas globales, con regulación de las velocidades de ejecución de las tareas para alcanzar un nivel óptimo de utilización. Con esto evitan migrar tareas a otros hosts. Este trabajo proponen distintas políticas predictivas de calendarizacion de tareas globales para asignar dichas tareas en máquinas virtuales disponibles. Los resultados de la simulación muestran que este enfoque reduce el consumo total de energía de una nube.

La utilización $\mathrm{u_i(t)}$ del host i al tiempo t esta definida por:

$$u_i(t) = \frac{ah_i(t)}{mh_i}$$

Donde $ah_i(t)$ es la cantidad de MIPS asignadas al host i, y mh_i es el máximo poder de cómputo del host i. Los autores suponen que un host inactivo cambia su estado para apagarse inmediatamente. Por lo tanto, la potencia total de un host se define como:

$$P_i(u_i(t)) = \begin{cases} P_i^{static} + P_i^{dynamic}, & u_i(t) > 0 \\ 0 & ,en\ otro\ caso \end{cases}.$$

Donde $\mathrm{P_i^{static}}$ es la potencia consumida durante el tiempo de inactividad de un procesador y está definida como: $\mathrm{P_i^{static}} = \alpha \mathrm{P_i^{max}}$. $\mathrm{P_i^{max}}$ es la potencia consumida cuando un host trabaja a su máxima utilización, α

es la relación que existe entre la potencia estática de un host con respecto a su potencia máxima($0 < \alpha \leq 1$) la cual depende de las características físicas de cada host.

El consumo de energía dinámico se define como:

$$P_i^{dynamic} = \left(P_i^{max} - P_i^{static}\right) u_i(t).$$

Si el sistema consume la potencia $\mathrm{P(u)}$, la energía consumida será:

$$E = \int_0^{\frac{t_{min}}{u}} P(u)dt$$

Donde t_{min} es el tiempo en que un host trabaja a su máxima potencia de cómputo. Por lo tanto, el consumo de energía de un host se obtiene mediante:

$$E = [\alpha + (1 - \alpha)u]P^{max}\frac{t_{min}}{u}$$

DVMA- Enfoque de migración de máquina virtual descentralizada, por Wang et al. (2013). Los autores abordan un enfoque descentralizado de migración de máquinas virtuales. Definen un modelo de sistema y un modelo de energía basados en los vectores de carga, la recopilación de información de carga, la selección de VM y la determinación del destino.

El consumo de energía de un procesador se calcula de la siguiente manera:

$$P_i = \alpha * P_i^{max} + (1 - \alpha) * P_i^{max} * \theta$$

Donde P_i^{max} es el consumo de energía del nodo cuando se utiliza a su máxima capacidad (es decir, alcanza el 100% de la utilización del CPU). Además, α es la fracción de la potencia consumida por un nodo inactivo en comparación con un nodo a su máxima capacidad, y θ es la utilización actual de la CPU. El porcentaje de consumo de energía que depende de la utilización del CPU es similar al mostrado en la Figura 2.

EDRP- Calendarización de recursos consciente de la energía tomando en cuenta la fecha límite, por Gao et al. (2013). Los autores se centran en el problema de minimizar el costo de operación de un sistema en la nube al maximizar su eficiencia energética mientras aseguran que se cumplan las fechas límites definidas por los usuarios establecidas en el SLA. El algoritmo considera dos modelos de cargas de trabajo, solicitudes de conjuntos independientes y grafos de tareas con dependencias.

El modelo de consumo de potencia en el tiempo t, incluye el consumo de potencia estática $P^x_{static}(t)$ y el consumo de potencia dinámica $P^x_{dynamic}(t)$. Ambas potencias se encuentran correlacionados con el porcentaje de utilización del CPU $Util_x(t)$ en el tiempo t. La utilización en el tiempo t definida por $Util_x(t)$ únicamente considera los requerimientos de CPU que demandan las máquinas virtuales que se encuentran alojadas en el procesador. $P^x_{static}(t)$ es constante cuando $Util_x(t) > 0$, de lo contrario es cero. Los servidores tienen niveles óptimos de utilización Opt_x en términos de rendimiento por Watt.

Los autores parten de la suposición que para los servidores modernos el nivel óptimo de operación es $Opt_x \approx 0.7$. Si la utilización sobrepasa este nivel de operación, existe un aumento drástico en el consumo de energía en comparación con mantener el nivel de utilización por debajo de dicho punto $Util_x(t) < Opt_x$. Incluso para niveles de utilización idénticos, la eficiencia energética de los diferentes servidores puede variar. Esto es capturado por los coeficientes αx y βx, que representan el aumento de consumo de energía cuando $Util_x(t) < Opt_x$ y $Util_x(t) \geq Opt_x$, respectivamente. $P^x_{dynamic}(t)$ se calcula de la siguiente manera:

$$\begin{cases} Util_x(t)\alpha_x & \text{si } (Util_x(t) < Opt_x) \\ Opt_x * \alpha_x + (Util_x(t) - Opt_x)^2 \beta_x, & si\ (Util_x(t) \geq Opt_x) \end{cases}$$

Los autores señalan que la formulación exacta de $P^x_{dynamic}(t)$ no contradice el análisis presentado, ya que su incremento es mayor cuando $Util_x(t) \geq Opt_x$ que cuando $Util_x(t) < Opt_x$.

El consumo total de energía COSP es la suma del consumo de energía en todos los servidores a lo largo del intervalo de tiempo de la operación:

$$COSP = \sum_{x=1}^{M} \left(\sum_{t=1}^{L_{max}} \left(P^x_{static}(t) + P^x_{dynamic}(t) \right) \right)$$

Donde L_{max} es el límite superior de la duración máxima del calendario y M es el número de máquinas.

La Figura 4 muestra el consumo de energía en función de la utilización del CPU.

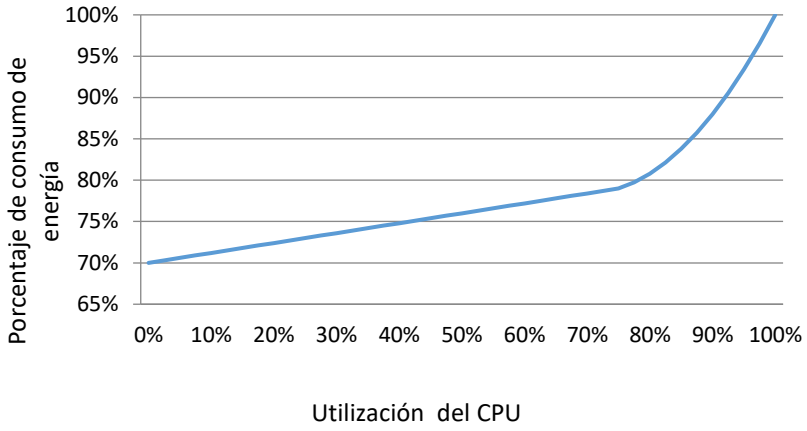


Figura 4. Función no lineal del consumo de energía (%) frente a la utilización del CPU (%).

BFDP-Decremento de potencia Best Fit, por Luo et al. (2013). Los autores proponen una metodología impulsada por simulación, utilizando un modelo de energía basado en la regresión polinómica LASSO (mínimo encogimiento absoluto y operador de selección). El algoritmo de calendarización de recursos BFDP tiene como objetivo mejorar la eficiencia energética sin degradar la calidad del servicio teniendo en cuenta cuatro tipos de trabajos: intensivos en CPU, intensivos en memoria, intensivos en red e intensivos en E / S. Se introduce un umbral de utilización para minimizar el problema de sobre consolidación en la estrategia Best-Fit. Los resultados mostraron que BFDP crea menos violaciones de SLA en cargas de trabajo ligeras.

El modelo de energía no lineal se define como:

$$y_i = \beta_0 + \sum_{j=1}^{m} \beta_j \phi_j(x_i) + \varepsilon_i,$$

Donde $\phi_j(x_i)$ es la función principal, x_i es la utilización de CPU y memoria. β_j es el parámetro determinado a través del modelo de proceso de entrenamiento. Finalmente, ε_i es una constante.

La Figura 5 presenta una relación que existe entre la utilización del CPU y la utilización de la memoria en la potencia total del sistema (L. Luo et al., 2013).

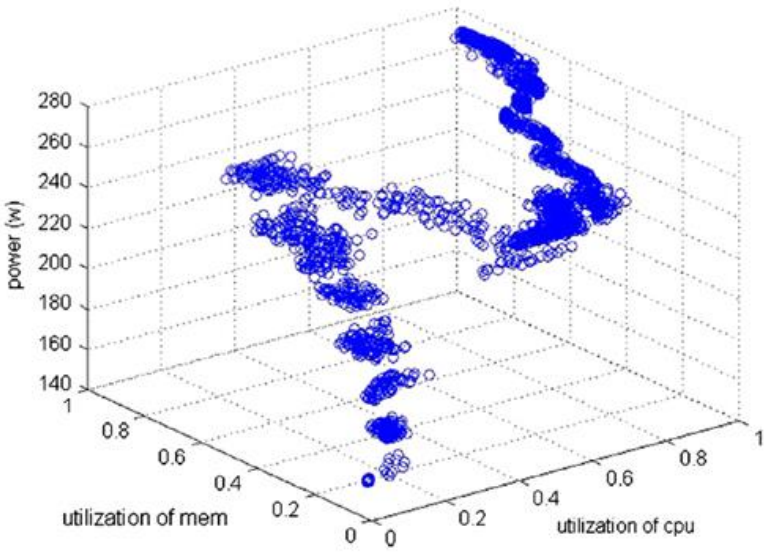


Figura 5. Consumo de energía en relación a la utilización de CPU y memoria**.**

PAHD- Implementación híbrida consciente de la energia, por Liu et al. (2010). Los autores consideran aplicaciones intensivas en E/S e intensivas en CPU para optimizar la utilización de recursos dentro de entornos virtualizados. En la evaluación experimental usan Xen para monitorear las máquinas virtuales. Los resultados experimentales, muestran un aumento en la eficiencia energética de un 2% a un 12% para diferentes configuraciones de asignación de recursos en comparación con la implementación predeterminada. Los autores concluyen que, si se asigna el doble de la capacidad de CPU a las aplicaciones intensivas en CPU en lugar de asignarla para aplicaciones intensivas en E/S, esto permite obtener una mejora significativa en la eficiencia energética.

En la Tabla 1, presentamos una lista de trabajos que utilizan algoritmos de asignación de recursos conscientes de la energía. La tabla incluye información sobre los dominios de la aplicación, las principales características de los algoritmos propuestos y los criterios utilizados para evaluar su desempeño.

Tabla 1. Características de algoritmos conscientes de la energía

	Ambiente		Características													Criterio de evaluación	
	Centro de datos	Nube	Centralizado	Descentralizado	En línea	Fuera de línea	Clarividente	No clarividente	Calidad de Servicio	Migración	Estático	Dinámico	Intensivo en CPU	Intensivo en E/S	Utilización	Energía	Fecha límite
EMVM		●	●		●			●	●	●		●	●		●	●	●
HSFL		●	●		●				●	●		●	●		●	●	
AETC		●	●			●	●		●	●			●		●	●	
CTES		●		●	●		●		●			●	●		●	●	●
DVMA		●		●					●	●			●		●	●	
EDRP		●	●			●	●		●	●	●		●		●	●	●
BFDP		●	●				●		●	●		●	●	●	●	●	
PAHD	●								●				●	●	●	●	

2.2 Algoritmos conscientes de la energía teniendo en cuenta los tipos de trabajo

En esta sección, presentamos algunos algoritmos de minimización de energía que consideran el tipo de aplicación en el proceso de asignación. Una de las principales diferencias entre los trabajos previos y esta tesis, es el modelo de energía, dado que considera la concentración de los tipos de trabajo, tanto en la estimación del consumo de energía, como en las estrategias de asignación.

VCTM- Personalización de máquinas virtuales y arquitectura de mapeo de tareas por Piraghaj et al. (2016). Los autores proponen una arquitectura para la asignación de recursos en la nube que asigna grupos de tareas a tipos de máquinas virtuales personalizadas. Su algoritmo utiliza patrones obtenidos del análisis de los datos históricos de utilización para asignar las tareas. El promedio de utilización de recursos U_T de una tarea T en términos de CPU, memoria y disco, que se obtiene usando la ecuación $U_T = \frac{\sum_{m=1}^{nr} u(T,m)}{nr}$. Donde, nr es el número de veces que se usó la tarea (u_T) el cual es reportado en los datos históricos y $u(T,m)$ es la $m - ésima$ observación del valor de utilización u_T en las trazas.

Su trabajo se centra principalmente en la utilización eficiente de los recursos de cómputo, e implementan una arquitectura de extremo a extremo para la asignación eficiente de trabajos en los centros de datos. El objetivo de este trabajo es reducir el consumo de energía de la infraestructura.

Los autores presentan un enfoque que clasifica las configuraciones de la máquina virtual, con respecto al CPU, la memoria y la capacidad del disco a través del agrupamiento de tareas. Su trabajo toma en consideración los patrones de uso de cada grupo. Además, proponen un enfoque que considera diferentes estimaciones, incluido el uso promedio de recursos de las tareas en cada clúster para la identificación de la capacidad de tareas en la VM. Esta capacidad es el número máximo de tareas que se pueden acomodar en una máquina virtual

Los resultados que presentan, muestran que cuando los recursos se asignan en función de los patrones de uso, se puede lograr un importante ahorro de energía. Con respecto al consumo de energía, presentan un algoritmo llamado, “Asignación de recursos basados en la utilización” (URA), que supera en promedio a los otros cinco algoritmos presentados en su trabajo. Sin embargo, el algoritmo URA aumenta la tasa promedio de rechazo de tareas y produce demoras en la ejecución de las mismas.

PACM- Consolidación de máquina virtual basada en el rendimiento por Roytman et al. (2013). Los autores presentan un sistema que consolida las máquinas virtuales de modo que la degradación del rendimiento se encuentre dentro de un límite ajustable y minimice los recursos inactivos. Los autores determinan cuánto se degradará cada máquina virtual cuando se coloca en diferentes conjuntos de máquinas virtuales para realizar la consolidación, e identifican cuántas máquinas virtuales se pueden colocar en un servidor de modo que se mantenga el rendimiento requerido. Los autores presentan PACMan, un administrador de consolidación consciente del rendimiento, que minimiza el costo de los recursos, así como el consumo de energía o la cantidad de servidores utilizados.

Finalmente, los autores evalúan PACMan usando degradaciones medidas en aplicaciones SPEC CPU 2006. Para minimizar el desperdicio de recursos mientras se preserva el rendimiento, PACMan opera a un 10% de lo óptimo. PACMan ahorra más del 30% de energía en comparación con los esquemas de consolidación que no representan la interferencia y mejora el costo total de las operaciones en un 22%.

RSAC- Algoritmo de calendarización de recursos para cómputo en la nube basado en métodos eficientes de optimización de energía por Luo et al. (2012). Los autores estudian la relación que existe entre los componentes de la infraestructura y el consumo de energía en el entorno de computación en la nube. Analizan la relación entre los tipos de tareas y los métodos de ajuste de potencia de los componentes. El algoritmo clasifica los recursos de cómputo en cuatro categorías: CPU, memoria, almacenamiento y red.

El siguiente paso del es algoritmo es crear las plantillas correspondientes según el tipo de trabajo para cambiar el método que utilizan los componentes del clúster y así, lograr el objetivo de ahorro energía. Los autores diseñan diversas estrategias de regulación para diferentes componentes, y proponen un algoritmo de programación dinámica de recursos basado en la optimización energética del CPU, la memoria principal y el almacenamiento. Para evaluar el desempeño del enfoque propuesto, utilizan un entorno de computación en la nube real. Para controlar el consumo de energía del servidor, eligieron el método de medición indirecta mediante sensores integrados en el servidor, para medir el consumo de energía del servidor en tiempo real. En sus resultados experimentales, los autores demuestran que para los trabajos que no utilizan completamente el entorno de hardware, el uso del algoritmo propuesto puede reducir significativamente el consumo de energía.

2.3 Calendarización consciente de la contención

Los siguientes trabajos consideran la contención en el proceso de asignación de recursos.

TCAS- Calendarización consciente de la contención (Blagodurov y Fedorova, 2012). Los autores diseñaron, implementaron y evaluaron un sistema HPC de clúster virtualizado consciente de la contención. El objetivo de su investigación es detectar y mitigar los efectos de contención en la memoria (tasa de error de caché de último nivel) en los nodos del clúster. Además, se identifica sobre la marcha aquellos nodos que sufren contención de recursos compartidos, con un método robusto y no intrusivo de creación de perfiles. El problema se enfoca a nivel de los nodos mediante la programación de cada nodo del clúster y considera a nivel del clúster a través de la migración en vivo de la carga de trabajo del clúster MPI desde los servidores contendidos. La atenuación de la contención de recursos compartidos genera mejores tiempos de ejecución para la carga de trabajo del clúster.

ASCAR- Sistema automático de reducción de contención en el almacenamiento (Li et al., 2015). Los autores implementan un sistema llamado ASCAR para aumentar la utilización del ancho de banda y reducir la variación de velocidad. Llevan a cabo una serie de experimentos para comprender la similitud entre las cargas de trabajo con la aplicación de reglas de control de tráfico. Finalmente, su sistema puede mejorar el rendimiento de todas las cargas de trabajo probadas, hasta en un 35%. ASCAR mejora el rendimiento de una carga de trabajo NPB BTIO de la NASA (un benchmark de E/S realista) en un 33.5% y reduce la variación de velocidad en un 55.4%.

IIAC- Impacto de la contención entre aplicaciones por (Jokanovic et al., 2010). Los autores evalúan el impacto de la contención entre aplicaciones utilizando cargas de trabajo reales de HPC bajo diferentes esquemas de ruteo en árboles densos reducidos (slimmed fat trees). El porcentaje de pérdida en el rendimiento del sistema debido a la contención entre aplicaciones en los sistemas de HPC actuales es de aproximadamente un 10%. Asimismo, presentan una proyección del impacto de la contención entre aplicaciones en los sistemas HPC futuros a corto y mediano plazo, al escalar el poder computacional del nodo y las velocidades del enlace de red a valores previsibles. Su proyección para sistemas de HPC muestra que la contención entre aplicaciones puede causar una pérdida en el rendimiento del 15% incluso con velocidades de enlace de 40 Gb/s para algunas mezclas de aplicaciones. Finalmente, encontraron una alta correlación entre el volumen de comunicación de las aplicaciones en una carga de trabajo y la cantidad de contención entre aplicaciones.

CCAP- Asignación de máquina virtual consciente de la contención en caché (Jin et al., 2015). Los autores analizan la interferencia en el rendimiento debido a la disputa por SLLC (Share Last Level Cache) en la nube de HPC. Emplean una técnica de análisis de distancia de reutilización mejorada con un algoritmo de compresión cíclica acelerada. Con el objetivo de identificar la intensidad de interferencia de caché de la aplicación, cuando varias VM que tienen aplicaciones HPC virtuales se ejecutan por separado en diferentes núcleos físicos. El problema de la contención SLLC también surge y disminuyendo la eficiencia entre las máquinas virtuales que se encuentran en núcleos que pertenecen al mismo chip. Además, clasifican las aplicaciones HPC en tres categorías: programas de contaminación de caché, programas sensibles a la caché y programas amigables con la caché.Los autores proponen una solución de software práctica, colocación de máquina virtual con conciencia de caché (CCAP), para abordar la interferencia de rendimiento debido a la contención de SLLC en la nube de HPC.

La evaluación de CCAP mejora significativamente el rendimiento de las aplicaciones sensibles a la caché y reduce el tiempo de ejecución en un 12% y la tasa de fallas de la memoria caché en un 13%, además de aumentar el rendimiento en un 13% en promedio. Para las aplicaciones de contaminación de caché, CCAP también tiene una mejora del 5% en el rendimiento.

Kyoto- Aplicando el principio de pagar por contaminadores a la contención del caché (Bui et al., 2011).Los autores proponen Kyoto, una solución de software con el objetivo de minimizar el impacto en el rendimiento de la VM debido a la contención LLC (Last Level Cache). Se basan en los contadores del monitoreo del rendimiento del hardware (por sus siglas en inglés PMC) para supervisar la actividad en el caché de las máquinas virtuales y así medir el nivel de contaminación de la memoria caché. A cada máquina virtual se le asigna un nivel de contaminación permitido. Una máquina virtual, que supera la contaminación

permitida, tiene por consecuencia una reducción en la capacidad del CPU. En su investigación, implementaron un nuevo planificador vCPU, que al iniciar aplica una reservación del nivel de contaminación por máquina virtual. El prototipo implementado se evaluó utilizando el bechmark (SPEC CPU2006), lo que demuestra que puede imponer el aislamiento del rendimiento entre máquinas virtuales, incluso en caso de contención LLC.

2.4 Monitoreo de energía

Para obtener un perfil detallado del consumo de energía de un servidor, se puede implementar un dispositivo de monitoreo en cada *rack* o servidor. La información recabada se puede utilizar para desarrollar estrategias de calendarización para la optimización energética. En este trabajo se perfiló el consumo de energía de los diferentes componentes requeridos por un tipo de trabajo. A continuación, se presentan los trabajos relacionados con el monitoreo de energía en un centro de datos.

Bellosa (2000) presenta uno de los primeros modelos que correlaciona los contadores de hardware (hardware counters) con el consumo de energía. Los autores proponen que existe una fuerte correlación entre el consumo de energía y las operaciones sobre enteros y de punto flotante, así como las solicitudes de memoria debidas a errores de caché. Para este análisis, ejecutan *benchmarks* durante varios segundos, y miden el consumo de potencia y la energía de todo el sistema utilizando un multímetro.

NIPD- Desagregación de potencia no intrusiva por Tang et al. (2015). Los autores presentan una solución sin costo agregado, puramente basada en software para la supervisión de energía detallada a nivel del servidor. Utilizan una técnica de desagregación de energía no intrusiva (NIPD), que establece funciones de asignación de potencia (PMF) entre los estados de los servidores y su consumo de energía. Con esto, infieren el consumo de energía de cada servidor con la potencia agregada de todo el centro de datos al ejecutar diferentes benchmarks. Recolectan datos de entrenamiento y actualizan las PMF. después de cada actualización de las PMF, ejecutan cargas de trabajo sintéticas (compilador, inteligencia artificial: go and chess, simulación de eventos discretos) y recopilan el consumo de energía del rack y los estados de los componentes de servidor, y calculan la estimación de potencia con las PMF actualizadas. Sus resultados muestran que NIPD proporciona una estimación de potencia de alta precisión a nivel del rack, con un error relativo medio de 2.63%. A nivel de servidor, con un error relativo medio de 10.27% y 8.17% para la estimación de potencia inactiva y pico de potencia, respectivamente.

ECMS- Consumo de energía de los servidores multinúcleo basados en ARM por Tudor et al. (2013).Los autores presentan un modelo de energía que predice el consumo como un factor que depende del número de núcleos y la frecuencia del reloj central. El enfoque consiste en dividir la potencia consumida en función de los tres tipos de recursos: núcleos, memoria y dispositivo de E/S para comprender el rendimiento energético de las cargas de trabajo ejecutadas en un servidor multinúcleo. Los autores observaron que los servidores multinúcleo de baja potencia, no siempre ofrecen ejecuciones energéticamente eficientes para las cargas de trabajo. Esto se debe a los grandes desequilibrios entre los núcleos, la memoria y los recursos de E/S, lo cual pueden conducir a recursos subutilizados y contribuir así al desperdicio de energía.

Capítulo 3. Modelado de energía

3.1 Modelo de energía

El consumo de energía se divide en dos componentes: potencia estática y potencia dinámica. La potencia estática se consume cuando el componente no está activo o en funcionamiento. La potencia estática también se conoce como potencia inactiva o potencia base (e_{idle}). El consumo dinámico de energía se produce cuando una aplicación utiliza el componente para su ejecución (e_{used}).

El modelado de energía se define como el estudio de las correlaciones lineales y no lineales entre la utilización del sistema y el consumo de energía. Hay diversas investigaciones que modelan los consumos de potencia y energía. Los trabajos relacionados (Beloglazov et al., 2012; J. Luo et al., 2014; Wang et al., 2013) consideran el uso de un modelo energético, basado en la observación de que la utilización de CPU está altamente correlacionada con el consumo total de energía de un servidor. Estos trabajos usan modelos lineales, que se basan en los niveles de utilización actuales para estimar el consumo de energía. Por otro lado, existen trabajos que consideran el uso de modelos no lineales (Gao et al., 2013; Hsu et al., 2014). En nuestro trabajo de investigación, consideramos el consumo de energía como una función que toma en cuenta la utilización del CPU y la concentración de trabajos del mismo tipo.

Seguimos el modelo híbrido no lineal de consumo de energía propuesto en (Armenta-Cano et al., 2017). En este modelo, consideramos un procesador caracterizado por su poder computacional s (número de operaciones por unidad de tiempo - MIPS), y n trabajos independientes $J_1, J_2, \dots, J_n$. El trabajo J_l es descrito por la tupla $J_l = (r_l, a_i, c_l)$, donde $r_l \geq 0$ es el tiempo cuando la tarea se encuentra lista para ser ejecutada; el parámetro r_l de un trabajo no está disponible antes de que se envíe el trabajo. c_l es la capacidad de cómputo medida en millones de instrucciones por segundo (MIPS) garantizada para la aplicación. La utilización $u_l(t)$ es la contribución que aporta el trabajo J_l al procesador al tiempo t y es definido como: $u_l(t) = \frac{c_l(t)}{s}$, donde $c_l(t)$ es la capacidad de cómputo garantizada para la aplicación en el momento t. Finalmente, $a_i \in \mathrm{A} = \{a_1, a_2, \dots, a_k\}$, caracteriza un tipo de trabajo como intensivo en CPU (CI), intensivo en memoria (MI), intensivo en disco (DI), entre otros. La cardinalidad de A es la cantidad de diferentes tipos de trabajo.

El consumo de potencia $e(t)$ del procesador al tiempo t consiste de dos partes: e_{idle} (el procesador está encendido, pero no se usa) y $e_{used}(t)$ (el procesador está encendido y ejecuta trabajos):

$$e(t) = o(t)(e_{idle} + e_{used}(t)) \quad (1)$$

Donde $o(t) = 1$, si el procesador se encuentra encendido al tiempo t, y $o(t) = 0$, de otra forma.

Sea $U_{a_i}(t) = \sum_{j=1}^{m} u_j(t)$ la utilización que aportan m trabajos de tipo a_i al procesador. Se define la fracción de consumo de potencia cuando el procesador ejecuta el tipo de trabajo a_i por f_{a_i}, y se calcula de la siguiente forma:

$$f_{a_i}\left(U_{a_i}(t)\right) \quad \mathrm{a}_i \in \mathrm{A} \tag{2}$$

Calculamos $F(t)$ como la suma de la porción de potencia consumida por cada tipo de trabajo de la siguiente manera:

$$F(t) = \sum_{\forall\, a_i \in \mathrm{A}} f_{a_i}\left(U_{a_i}(t)\right), \quad 0 \leq F(t) \leq 1 \tag{3}$$

Tomando en cuenta la proporción de uso de CPU que contribuye cada aplicación, calculamos el consumo de potencia $e_{used}(t)$, considerando los siguientes parámetros:

Primero, se obtiene la utilización total de CPU en el tiempo t por:

$$U_T(t) = \sum_{\forall\, a_i \in \mathrm{A}} U_{a_i}(t) \tag{4}$$

La combinación de aplicaciones se considera en base a la concentración de cada aplicación, la cual se define de la siguiente forma.

Sea $\alpha_{a_i}(t) = \frac{U_{a_i}(t)}{U_T(t)}$ la concentración (proporción) de trabajos de tipo a_i entre todos los trabajos que se están ejecutando en el procesador en el tiempo t. Si consideramos sólo dos tipos de aplicaciones, CI y MI. Entonces: $\mathrm{A} = \{CI, MI\}$, y $\alpha_{CI}(t) + \alpha_{MI}(t) = 1$.

Finalmente, el consumo de potencia $e_{used}(t)$ se calcula de la siguiente forma:

$$e_{used}(t) = (e_{max} - e_{idle}) * F(t) * g\left(\alpha_{a_i}(t)\right) \tag{5}$$

Donde e_{max} es el consumo de potencia máximo cuando el procesador se utiliza por completo. La función $g\left(\alpha_{a_i}(t)\right)$, representa la fracción de consumo de potencia cuando dos o más aplicaciones se ejecutan en el mismo procesador. Suponemos 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. Figura 12 y Figura 13 muestran los perfiles energéticos y de potencia de las aplicaciones CI y MI frente a la

utilización del CPU, tanto a nivel de procesador como de servidor. Para más detalles, ver la sección 3.3 (Figura 16).

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, $g\left(\alpha_{a_i}(t)\right) = 1$.

Por último, se define al consumo total de energía como la integral de la potencia consumida durante el máximo tiempo de operación C_{max}:

$$E^{op} = \int_{t=1}^{C_{max}} e(t)dt \tag{6}$$

3.2 Metodología para la medición de energía

En esta sección, analizamos las mediciones del consumo de potencia y energía de diferentes tipos de trabajos. Llevamos a cabo un conjunto de evaluaciones empíricas para definir un modelo de consumo de energía. En la primera fase, medimos y comparamos la potencia y el consumo de energía de cada tipo de trabajo bajo diferentes niveles de utilización. En la segunda fase, medimos la potencia y el consumo de energía a diferentes niveles de utilización del CPU, ejecutando la combinación de diferentes tipos de trabajos.

La estimación del consumo de energía del servidor a diferentes niveles de utilización y concentración, se realiza en función del modelo energético propuesto en la presente investigación. Es esencial explicar la terminología de potencia y energía para comprender sus mecanismos de medición. La potencia se puede definir como la relación de paso de energía de un flujo por unidad de tiempo. La potencia se mide en Joules/segundos (Watts). Mientras, la energía es la cantidad total de trabajo realizado durante un período (es la integral de la potencia en el tiempo), y se mide en Joules.

$$Potencia\ (W) = \frac{trabajo\ (J)}{Tiempo\ (seg)}$$

$$Energía\ (J) = potencia\ \times Tiempo$$

Un kilowatt-hora es el equivalente a la potencia de 1,000 Watts que se aplica durante una hora.

$$Energía\ (kWh) = \frac{Energía\ (J)}{1000(J/seg)\ \times 3600(\text{seg})}$$

Por un lado, para realizar las mediciones del consumo de potencia y energía a nivel de procesador, utilizamos likwid-powermeter (Treibig et al., 2010), una herramienta que permite el acceso al contador RAPL (Running Average Power Limit) en los procesadores Intel.

Por otro lado, utilizamos una unidad especializada en distribución de energía (PDU) que mide el consumo total de energía del servidor (“Switched PDU with Per Outlet Power Metering Outlet Metered Reboot Switch,” 2017).

Independientemente del nivel donde se realiza la medición, ya sea con likwid-powermeter o PDU, asignamos una aplicación por núcleo y no usamos los núcleos lógicos del servidor. Las pruebas consisten en asignar cada aplicación a un núcleo específico y obtener la medición de energía durante su ejecución. En seguida, incrementamos en uno la cantidad de trabajos hasta que se alcanza la máxima utilización del procesador; desde un sólo núcleo hasta un máximo de ocho núcleos.

En las siguientes subsecciones, presentamos más detalles de las herramientas utilizadas para las mediciones de potencia y energía. Además, explicamos los *benchmarks* utilizados como tipos de trabajo.

3.2.1 LIKWID

Para analizar el consumo de potencia y energía del procesador, utilizamos LIKWID ("Like I Knew What I am Doing "). LIKWID es un conjunto de herramientas orientadas al rendimiento del sistema.

LIKWID se maneja por medio de línea de comandos. Está orientado a la programación ubicada al rendimiento en un entorno Linux, no requiere ninguna revisión del kernel, y es adecuado para las arquitecturas de procesadores Intel y AMD (Treibig et al., 2010).

Para las mediciones utilizamos tres herramientas de LIKWID:

Likwid-topology

Las computadoras modernas contienen múltiples núcleos; y cada núcleo es capaz de procesar múltiples hilos. Es crucial asignar los hilos a núcleos dedicados, en especial, si se tiene caché y memoria principal compartidas entre los núcleos. Es esencial conocer la topología del hardware para comprender las propiedades de rendimiento de un nodo. La herramienta likwid-topology extrae esta información de la biblioteca *hwloc* o de la instrucción *CPU id*.

Likwid-topology puede desplegar la siguiente información del servidor:

- En el hilo, se muestra cómo se asignan los identificadores (IDs) de los procesadores a los recursos computacionales físicos
- En el caché, se muestra cómo los procesadores comparten la jerarquía de caché
- En las propiedades del caché, se reporta información detallada de todos los niveles de memoria caché
- En la topología NUMA, se reporta todos los dominios NUMA y tamaños de memoria

Máquinas con hardware idéntico pueden etiquetar los hilos de manera diferente debido al BIOS. Para las aplicaciones que utilizan hilos en un procesador de múltiples núcleos; es esencial asignar los hilos a los núcleos dedicados. Mientras el kernel de Linux ofrece un API para asignar los hilos, para poder implementar soluciones flexibles es necesario cierta codificación para poder tener afinidad. En el caso de procesadores Intel, se incluye un mecanismo de asignación sofisticado para implementar OpenMP. Sin embargo, esta operación es complicada dado que es controlada por variables del entorno.

En la Figura 6 y Figura 7 se muestran respectivamente las topologías de hilos y de caché del servidor que utilizamos para las mediciones.

Likwid-pin

Las implementaciones de OpenMP dependen de la API pthread. No obstante, se requiere de una solución de compilación independiente y una plataforma simple. Likwid-pin es una herramienta que puede pre-cargar una librería que contiene la llamada a pthread_create. En esta librería, los hilos son asignados utilizando Linux OS API. Likwid-pin tambien puede ser utilizada para asignar aplicaciones de forma serial como reemplazo a conjuntos de tareas (Treibig et al., 2010).

Likwid-pin ofrece tres diferentes sintaxis para especificar cómo se asignan los hilos a los procesadores:

- Utilizando una lista de hilos
- Especificando una expresión en base a la lista de hilos
- Utilizando políticas de dispersión

Utilizamos likwid-pin para forzar cada benchmark a un núcleo específico. Likwid-pin requiere de la información correcta acerca de la numeración de los hilos y la topología del caché, los cuales pueden ser obtenidos con likwid-topology.

```
--------------------------------------------------------------------------------
CPU name:       Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz
CPU type:       Intel Xeon IvyBridge EN/EP/EX processor
CPU stepping:   4
********************************************************************************
Hardware Thread Topology
********************************************************************************
Sockets:                2
Cores per socket:       8
Threads per core:       2
--------------------------------------------------------------------------------
HWThread        Thread          Core            Socket          Available
0               0               0               0               *
1               0               1               0               *
2               0               2               0               *
3               0               3               0               *
4               0               4               0               *
5               0               5               0               *
6               0               6               0               *
7               0               7               0               *
8               0               0               1               *
9               0               1               1               *
10              0               2               1               *
11              0               3               1               *
12              0               4               1               *
13              0               5               1               *
14              0               6               1               *
15              0               7               1               *
16              1               0               0               *
17              1               1               0               *
18              1               2               0               *
19              1               3               0               *
20              1               4               0               *
21              1               5               0               *
22              1               6               0               *
23              1               7               0               *
24              1               0               1               *
25              1               1               1               *
26              1               2               1               *
27              1               3               1               *
28              1               4               1               *
29              1               5               1               *
30              1               6               1               *
31              1               7               1               *
--------------------------------------------------------------------------------
Socket 0:               ( 0 16 1 17 2 18 3 19 4 20 5 21 6 22 7 23 )
Socket 1:               ( 8 24 9 25 10 26 11 27 12 28 13 29 14 30 15 31 )
--------------------------------------------------------------------------------
```

Figura 6. Topología de los hilos

```
********************************************************************************
Cache Topology
********************************************************************************
Level:                  1
Size:                   32 kB
Cache groups:           ( 0 16 ) ( 1 17 ) ( 2 18 ) ( 3 19 ) ( 4 20 ) ( 5 21 ) ( 6 22 )
( 7 23 ) ( 8 24 ) ( 9 25 ) ( 10 26 ) ( 11 27 ) ( 12 28 ) ( 13 29 ) ( 14 30 ) ( 15 31 )
--------------------------------------------------------------------------------
Level:                  2
Size:                   256 kB
Cache groups:           ( 0 16 ) ( 1 17 ) ( 2 18 ) ( 3 19 ) ( 4 20 ) ( 5 21 ) ( 6 22 )
( 7 23 ) ( 8 24 ) ( 9 25 ) ( 10 26 ) ( 11 27 ) ( 12 28 ) ( 13 29 ) ( 14 30 ) ( 15 31 )
--------------------------------------------------------------------------------
Level:                  3
Size:                   20 MB
Cache groups:           ( 0 16 1 17 2 18 3 19 4 20 5 21 6 22 7 23 )
                        ( 8 24 9 25 10 26 11 27 12 28 13 29 14 30 15 31 )
--------------------------------------------------------------------------------
********************************************************************************
NUMA Topology
********************************************************************************
NUMA domains:           2
--------------------------------------------------------------------------------
Domain:                 0
Processors:             ( 0 16 1 17 2 18 3 19 4 20 5 21 6 22 7 23 )
Distances:              10 11
Free memory:            8338.66 MB
Total memory:           32579.2 MB
--------------------------------------------------------------------------------
Domain:                 1
Processors:             ( 8 24 9 25 10 26 11 27 12 28 13 29 14 30 15 31 )
Distances:              11 10
Free memory:            6967.13 MB
Total memory:           32768 MB
--------------------------------------------------------------------------------
```

Figura 7. Topología de la memoria cache

Likwid-powermeter

Likwid-powermeter es una herramienta que permite consultar el consumo de potencia dentro de un paquete durante un periodo determinado. Con esta información, se calcula el consumo de energía final de los componentes de nivel de CPU enumerados en la Tabla 2, (Treibig et al., 2010). Likwid-powermeter permite el acceso a los contadores RAPL en el procesador Intel. Los procesadores Intel introdujeron con la arquitectura SandyBridge la interfaz RAPL para configurar y leer el consumo de energía de los procesadores y la memoria. La interfaz RAPL se controla a través de registros MSR. Esta interfaz permite consultar la energía consumida dentro de un paquete durante un período determinado y calcula el consumo de energía total resultante.

Tabla 2. Lista de sensores RAPL

RAPL_PKG	Paquete completo de CPU
RAPL PP0	Núcleos del procesador solamente
RAPL PP1	“Un dispositivo específico en el uncore”
RAPL DRAM	Controlador de memoria

La Figura 8, muestra un ejemplo de la salida de likwid-powermeter cuando se ejecuta una aplicación de CI durante 60 segundos en un núcleo. Utilizamos likwid-pin para asignar una aplicación de CI al núcleo ocho. Las métricas Energía [J] y Potencia [W] presentan la cantidad total de energía (Joules) y potencia (Watts), respectivamente. La métrica de tiempo de ejecución (RDTSC) [s] muestra todo el tiempo de ejecución en segundos.

```
+----------------------------+------------+-----------+-----------+-----------+
|           Metric           |     Sum    |    Min    |    Max    |    Avg    |
+----------------------------+------------+-----------+-----------+-----------+
|  Runtime (RDTSC) [s] STAT  |   978.0544 |   61.1284 |   61.1284 |   61.1284 |
| Runtime unhalted [s] STAT  |   160.8996 |    0.2842 |   76.7744 |   10.0562 |
|      Clock [MHz] STAT      | 48602.1254 | 3000.5291 | 3283.3347 | 3037.6328 |
|  Uncore Clock [MHz] STAT   |  3278.2465 |         0 | 3278.2465 |  204.8904 |
|          CPI STAT          |    23.5342 |    0.5239 |    3.5252 |    1.4709 |
|      Energy [J] STAT       |  3677.7676 |         0 | 3677.7676 |  229.8605 |
|       Power [W] STAT       |    60.1646 |         0 |   60.1646 |    3.7603 |
+----------------------------+------------+-----------+-----------+-----------+
```

Figura 8. Archivo de salida al ejecutar la aplicación CI utilizando likwid-powermeter

3.2.2 PDU

Una técnica convencional para la medición de potencia y energía en un centro de datos consiste en recopilar los datos obtenidos mediante PDUs conectados a los equipos de TI. El hardware dedicado del PDU proporciona datos precisos sobre el consumo de energía. Para las mediciones a nivel del servidor, utilizamos un PDU VMR-8HD20-1, Figura 9. Solamente se tiene conectado al PDU el servidor que está siendo evaluado. (“Switched PDU with Per Outlet Power Metering Outlet Metered Reboot Switch,” 2017).

Figura 9. Medidor de salida VMR-8HD20-1

El PDU con la capacidad de medir potencia, incluye una selección robusta de características de monitoreo, incluidas alarmas de eventos, medición específica de salida. La Figura 10 representa un ejemplo de conexiones de PDU con dispositivos de TI y sus pantallas.

A continuación, enlistamos algunas características del PDU VMR-8HD20-1 (“Switched PDU with Per Outlet Power Metering Outlet Metered Reboot Switch,” 2017).

Control y administración de energía:

- Conmutación remota de salida tanto individual como grupal
- Conmutación de Encendido/Apagado/Reiniciar/Predeterminado/Reducción de carga
- Secuencia en los enchufes y retardo en el encendido
- Estado predeterminado del encendido definido por el usuario
- Retención del estado del enchufe
- Etiquetar cada enchufe, agrupar y control de acceso
- Programado de encendido / apagado / reinicio
- Ping perro guardián con reinicio automático

Medición / Monitoreo / Informes:

- Medición del valor actual RMS
- Medición de toma de corriente, grupo y por ramas
- Informes de kWh, kW, Amps, Volts, Temperatura
- Alarmas de sobrecorreinte
- Alarma de enchufe con altas y bajas corriente
- Alarmas de sobretemperatura
- Alarma de perdida de voltaje (entrada de la línea)
- Alarma de ping sin respuesta
- Alarma de bloqueo de acceso inválido
- Alarma de ciclo de energía
- Alarma de Interruptor de circuito abierto
- Alarma sin de tono al marcar
- Notificación de alarma por correo electrónico, SNMP, SYSLOG
- Alarma, comando, auditoría de acceso, corriente, voltaje, watts y registro de temperatura

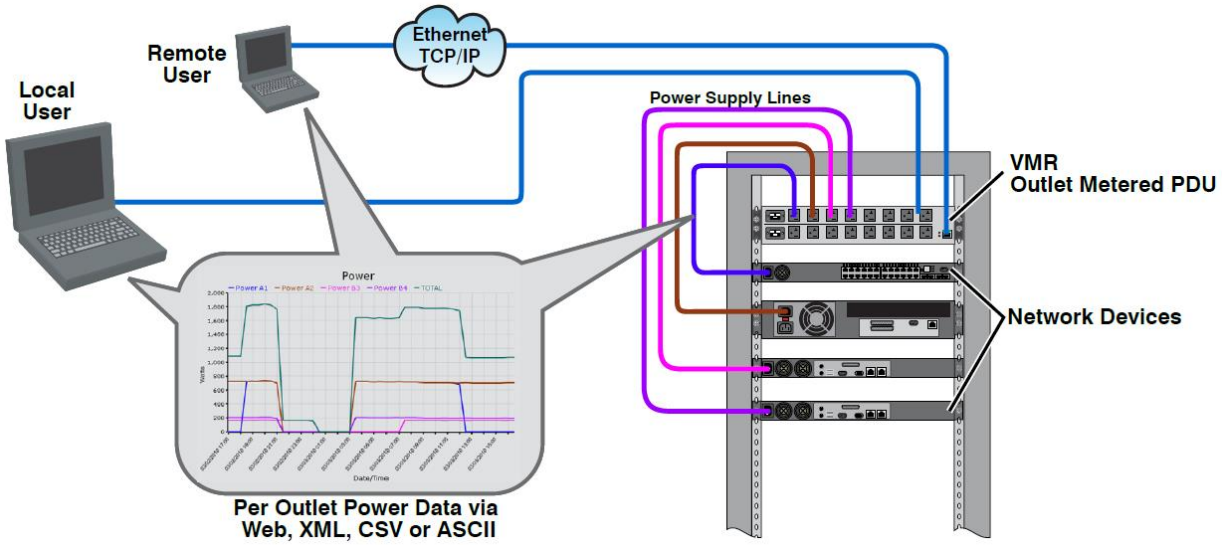
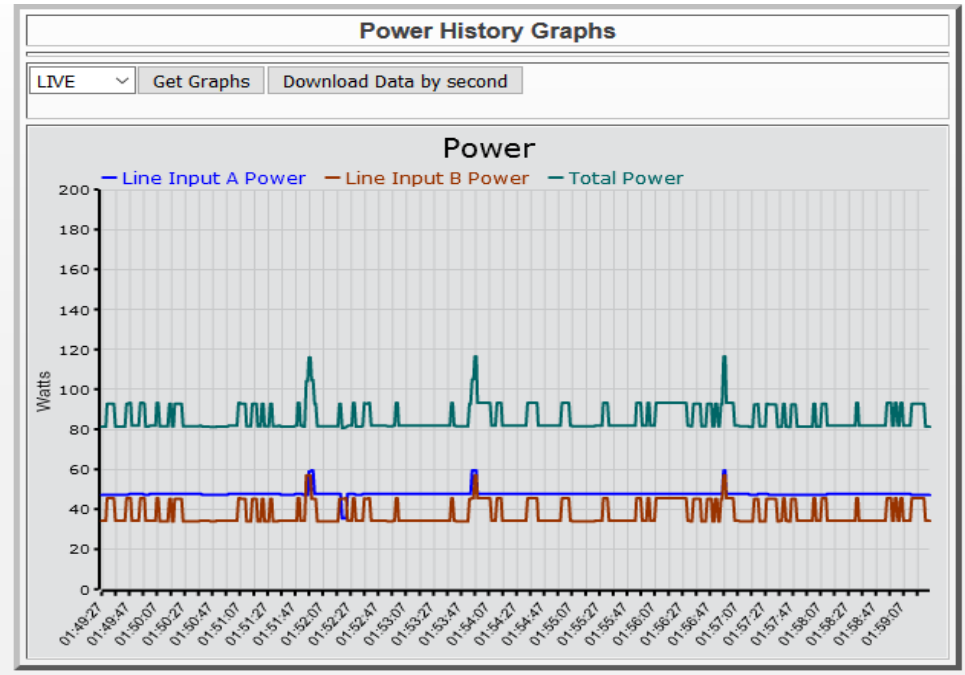


Figura 10. Esquema de conexiones al PDU con medición de potencia en cada enchufe

Para recopilar todas las mediciones del PDU, implementamos un script usando JavaScript con la herramienta Greasemonkey. Con el uso del script, accedemos al menú "Medición / Medición de potencia / Historial de energía" en la interfaz web con el firmware, versión 1.64 y descargamos todos los datos recopilados en formato HTML, consulte la Figura 11.

El PDU sólo presenta información sobre la potencia (Watts). Por lo tanto, necesitamos calcular el consumo de energía (Joules) de la siguiente manera; recopilamos los datos de mediciones de potencia; calculamos la potencia promedio consumida de cada ejecución y la multiplicamos por el tiempo total de ejecución de cada prueba.



Hour	Minute	Seconds	Line A	Line B	Line Total
1	49	27	47.2	34.2	81.4
1	49	28	47.2	34.2	81.4
1	49	29	47.2	34.2	81.4
1	49	30	47.2	34.2	81.4
1	49	31	47.2	45.6	92.8
1	49	32	47.2	45.6	92.8
1	49	33	47.2	45.6	92.8
1	49	34	47.2	45.6	92.8
1	49	35	47.2	45.6	92.8
1	49	36	47.2	45.6	92.8
1	49	37	47.2	34.2	81.4
1	49	38	47.2	34.2	81.4
1	49	39	47.2	34.2	81.4

Figura 11. Datos recopilados en watts

3.2.3 Benchmarks

Para simular los diferentes tipos de aplicaciones, utilizamos *benchmarks*, los cuales son un programa o conjunto de programas, que se emplean para evaluar el rendimiento de diferentes partes del sistema (CPU, memoria, disco, red), bajo ciertas condiciones de referencia. En el proceso de evaluación

comparativa, podemos recopilar métricas directamente del Sistema Bajo Pruebas (SUT), como el rendimiento o el consumo de energía.

En la Tabla 3, enumeramos algunos *benchmarks* que se utilizan en la literatura.

Tabla 3. Benchmarks

Benchmark	CPU	Memoria	E/S-Red	E/S-disco
LINPACK	●			
STREAM		●		
SPEC	●	●		
iperf			●	
IOR				●
IOzone				●
NPB	●	●		●
Ubench 0.32	●	●		
SysBench	●	●		●

LINPACK: es una colección de subrutinas para resolver sistemas de ecuaciones lineales. El software LINPACK resuelve el problema de descomposición usando álgebra lineal numérica. Un problema específico que involucra una matriz A, factores que descompone a la matriz A en un producto de matrices puras, bien estructuradas que pueden ser manipuladas para resolver el problema original. LINPACK tiene la capacidad de manejar diferentes tipos de matrices y diferentes tipos de datos y proporciona una amplia gama de opciones(Dongarra et al., 2003).

STREAM: tiene como objetivo, obtener el mejor ancho de banda de memoria posible usando kernels vectoriales simples. STREAM es un programa que mide el ancho de banda de memoria sostenible; en MB/s, y la velocidad de cálculo correspondiente para kernels vectoriales simples. El benchmark STREAM está específicamente diseñado para trabajar con conjuntos de datos mucho más grandes que la memoria caché disponible en cualquier sistema dado; por lo que los resultados son (presumiblemente) más indicativos del rendimiento de las extensas aplicaciones de estilo vectorial (McCalpin, 2005).

iPERF: este benchmark realiza mediciones de rendimiento de la red. IPERF activa mediciones del ancho de banda máximo alcanzable en redes IP. Soporta el ajuste de varios parámetros relacionados con el tiempo,

los búferes y los protocolos: TCP, UDP, SCTP con IPv4 e IPv6. Para cada prueba, informa el ancho de banda, la pérdida y otros parámetros (Dugan et al., n.d.).

IOR: realiza pruebas comparativas de lectura y escritura del rendimiento POSIX de un SSD local. Para ambas operaciones, descarta cachés antes de leer o escribir 10GB de datos usando un tamaño de transferencia de 256KB y 10GB para el tamaño de bloque (Loewe et al., 2012).

IOzone: este benchmark es una herramienta de sistema de archivos. Produce y mide una variedad de operaciones de archivos. Iozone se ha utilizado para evaluar muchas clases de máquinas y se ejecuta en diferentes sistemas operativos. Es útil para realizar un amplio análisis del sistema de archivos de la plataforma informática de un proveedor. Iozone prueba el rendimiento de E/S del archivo para las siguientes operaciones: leer, escribir, volver a leer, volver a escribir, leer hacia atrás, fread, fwrite, lectura aleatoria, pread, mmap, aio_read, aio_write (Norcott y Capps, 2003).

The NAS Parallel Benchmarks (NPB): es un conjunto de programas diseñados para evaluar el rendimiento de supercomputadoras paralelas. El conjunto de *benchmarks* se derivan de las aplicaciones de dinámica de fluidos computacional (CFD). Consisten en cinco núcleos y tres pseudoaplicaciones en la especificación original de "lápiz y papel" (NPB 1). Este benchmark se ha ampliado para incluir nuevos programas para malla adaptativa no estructurada, E/S paralelas, aplicaciones multizona y redes computacionales. Los tamaños de los problemas en NPB están predefinidos e indicados como clases diferentes. Las implementaciones de referencia de NPB están disponibles en modelos de programación de uso común como MPI y OpenMP (Bailey et al., 1993).

UBENCH: el benchmark Unix, proporciona una medida única de rendimiento para máquinas que ejecutan varios tipos de sistema operativo Unix. Esta versión de benchmark cuenta únicamente con pruebas para CPU(s) y memoria (“Ubench,” n.d.).

SPEC: proporciona mediciones de rendimiento que pueden usarse para comparar cargas de trabajo intensivas en cómputo en diferentes sistemas informáticos. SPEC CPU2006 contiene dos suites de referencia: CINT2006 se utiliza para medir y comparar el rendimiento de operaciones de enteros intensivas de CPU, y CFP2006 se utiliza para medir y analizar el rendimiento de operaciones de punto flotante intensivas de CPU (Dixit, 1991).

SysBench: es una herramienta de referencia multiplataforma, modular y de subprocesos múltiples para evaluar los parámetros del SO que son importantes para un sistema que ejecuta una base de datos bajo

carga intensiva. La idea de este benchmark es obtener rápidamente una impresión sobre el rendimiento del sistema sin configurar parámetros de referencia de bases de datos complejos. (Kopytov, 2017).

En esta tesis, elegimos los *benchmarks* proporcionados por SysBench para simular los tipos de aplicaciones. Ejecutamos y analizamos tres *benchmarks*. Para CPU-Intensivo (CI), cada solicitud consiste en el cálculo de números primos hasta un valor especificado por la opción max primes. Todos los cálculos se realizaron usando enteros de 64 bits. Cada núcleo ejecuta las solicitudes al mismo tiempo hasta que se excede el número total de solicitudes o el tiempo total de ejecución. Ejecutamos cada aplicación de CI con los siguientes parámetros:

Sysbench: --test = cpu --cpu-max-prime = 2000000 --num-threads = 1 --max-time = 60

La prueba de memoria intensiva (MI) está diseñada específicamente para trabajar con conjuntos de datos mucho más grandes que la memoria caché disponible en cualquier sistema dado. Los resultados son más indicativos del rendimiento de las aplicaciones de estilo vectorial. Esta prueba se puede usar para comparar lecturas o escrituras de memoria secuenciales. En los experimentos, ejecutamos cada aplicación de MI con los siguientes parámetros:

Sysbench: --test = memory --memory-block-size = 8000M --memory-total-size = 900G --num-threads = 1 --max-time = 60 run

Si el usuario desea alcanzar un cierto porcentaje de la memoria total, el valor del tamaño del bloque de memoria debe establecerse y depende de la memoria total del servidor. En nuestro caso, configuramos 8000M para lograr una utilización total de memoria del 12.5% y 8600M para alcanzar el 13.4% de la memoria total del servidor.

Finalmente, para el benchmark intensivo en disco (DI), Sysbench produce archivos de varios tipos de cargas trabajo de E/S. En su primera etapa, SysBench crea una cantidad específica de datos con un tamaño total determinado; luego, en la etapa de ejecución, cada subproceso realiza operaciones de E/S especificadas en este conjunto de archivos. SysBench realiza la validación de checksums en todos los datos leídos del disco. En cada operación de escritura, el bloque se llena con valores aleatorios, y la suma de verificación se calcula y almacena en el bloque junto con el desplazamiento de este bloque dentro de un archivo. En cada operación de lectura, el bloque se valida comparando el desplazamiento almacenado con el desplazamiento real y la suma de comprobación almacenada con la suma de control inicialmente calculada (Kopytov, 2017). Ejecutamos cada aplicación de DI con los siguientes parámetros:

Sysbench: --num-threads = 1 --test = fileio --file-total-size = 5G --file-test-mode = rndrw --max-time = 60

Los tres puntos de referencia se establecieron con un límite de tiempo de ejecución de 60 segundos.

3.3 Análisis experimental del consumo de energía

Analizamos el consumo de energía de la ejecución de diferentes tipos de aplicaciones en un servidor. El servidor que se utiliza en los experimentos es un Express x3650 M4, con dos procesadores Xeon IvyBridge E5-2650v2 95W, velocidad de reloj predeterminada de 2.6GHz. Cada procesador tiene ocho núcleos y dos hilos por núcleo (16 con hyper threading), memoria de nivel 1 de 32kB, nivel 2 de 256kB, nivel 3 de 20MB. Dos dominios NUMA de 32 GB cada uno, con una memoria total de 64 GB. Únicamente utilizamos ocho núcleos por procesador (no se utilizan núcleos lógicos). El sistema operativo del servidor es una versión Linux, CentOS 7.1.1503.

Cuando hacemos referencia al consumo de energía de un componente o una aplicación, nos referimos al consumo total; que es una suma de las potencias dinámicas y estáticas. Además, utilizamos indistintamente los términos trabajo/aplicación, a menos que se especifique lo contrario. Las mediciones reportan el consumo de potencia en Watts y el consumo de energía en Joules.

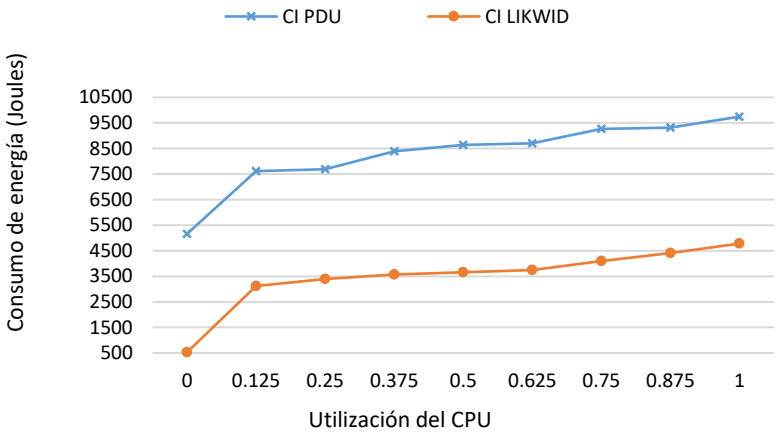
Para calcular el consumo de potencia inactiva, ejecutamos el comando de suspensión de Linux durante diferentes períodos a diferentes horas del día. En promedio, el consumo de potencia inactiva del procesador es de 8.8 W. Para calcular el consumo de potencia inactiva del servidor, recopilamos la información del PDU. El consumo de energía inactivo del servidor es de 86W en promedio.

Para obtener los perfiles de potencia y energía, recopilamos los datos de medición a diferentes niveles de utilización y concentración de CPU para cada tipo de aplicación. El servidor únicamente ejecuta las aplicaciones del sistema operativo y las aplicaciones que están siendo analizadas. Se obtuvieron el valor promedio de 20 ejecuciones por experimento.

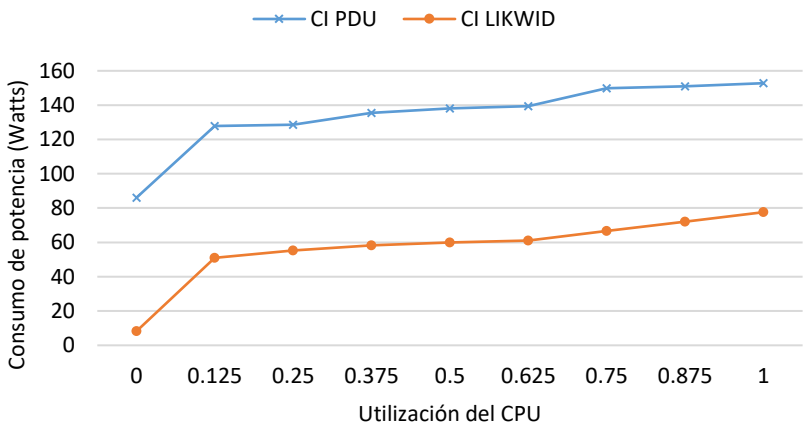
3.3.1 Perfiles energéticos de cada tipo de aplicación

Como se mencionó anteriormente, ejecutamos tres *benchmarks*, con un límite de tiempo de ejecución de 60 segundos. La primera aplicación que se analizó es de tipo CI. La Figura 12, muestra los perfiles

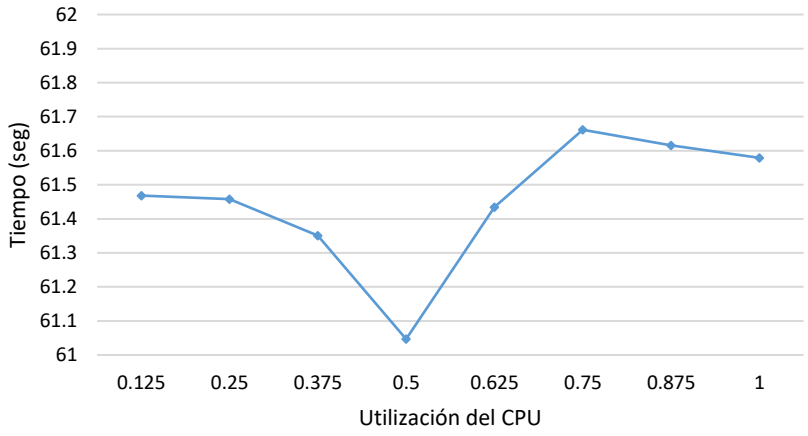
energéticos para aplicaciones de CI a diferentes niveles de utilización del CPU. Utilizamos tanto la herramienta LIKWID como el PDU. En la Figura 12a y Figura 12b podemos observar que las mediciones de PDU son aproximadamente 2.4x los valores obtenidos por LIKWID, excepto en modo inactivo, que en promedio es 10x. En la Figura 12c notamos que la diferencia del tiempo de ejecución es menor a un segundo cuando los niveles de utilización del CPU se incrementan.



(a) Consumo de energía de CI



(b) Consumo de potencia de CI



(c) Tiempo de ejecución de CI

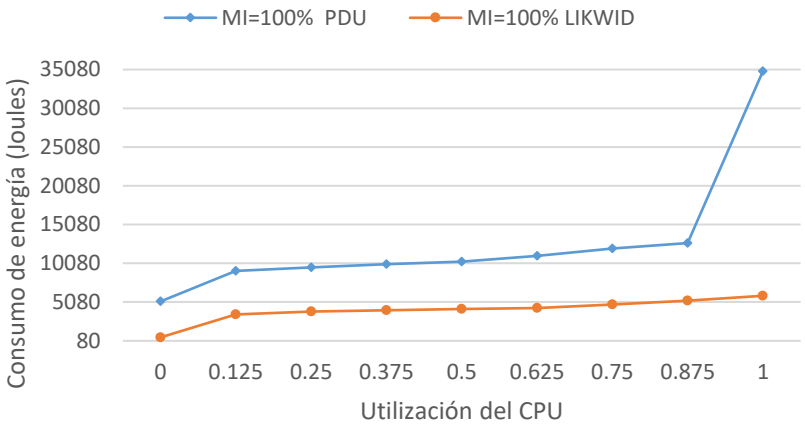
Figura 12. Perfiles de CI

La segunda aplicación que se analizó es de tipo MI. Para generar contención en memoria, ejecutamos ocho *benchmarks* de MI en el mismo procesador simultáneamente.

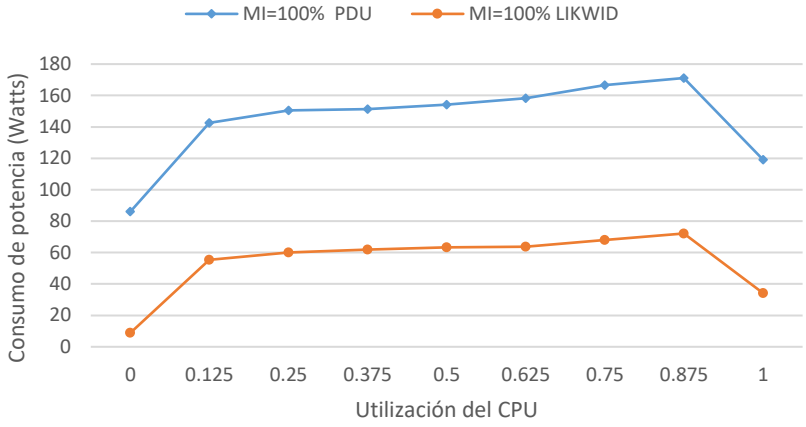
Primero, consideramos experimentos con MI = 100%. En este caso, la memoria del servidor está llena, es decir, las aplicaciones requieren el 100% de la memoria total del servidor. Se ejecutan 8 aplicaciones de tipo MI y cada una requiere el 12.5% de la memoria total del servidor.

En la segunda prueba, denominada MI = 107%, cada aplicación de tipo MI requiere un 13.4% de la memoria total del servidor. En este caso, la contención se presenta cuando se ejecutan 8 aplicaciones simultáneamente. Las aplicaciones requieren 107.2% de la memoria total del servidor.

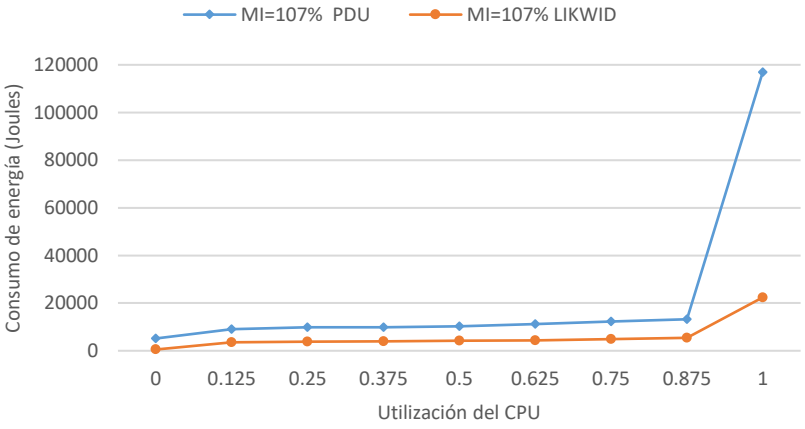
La Figura 13a y Figura 13c presentan los perfiles energéticos (Joules) al ejecutar las aplicaciones de MI cuando la utilización de memoria que se requiere es 100% y 107%, respectivamente. Debido a que se demanda un exceso de memoria en el servidor, se desencadena un alto consumo de energía. Esto se debe al incremento en el tiempo de ejecución (ver Figura 14). Adicionalmente, la Figura 13b y Figura 13d presentan el consumo de potencia (Watts). Debido a una condición de paginado, se produce una degradación en el rendimiento del sistema y el disco se utiliza como una extensión de la memoria al intercambiar datos rápidamente en entre la memoria y el disco. En esta situación, el consumo promedio de potencia disminuye. Sin embargo, el tiempo de ejecución del sistema aumenta significativamente, ver Figura 14.



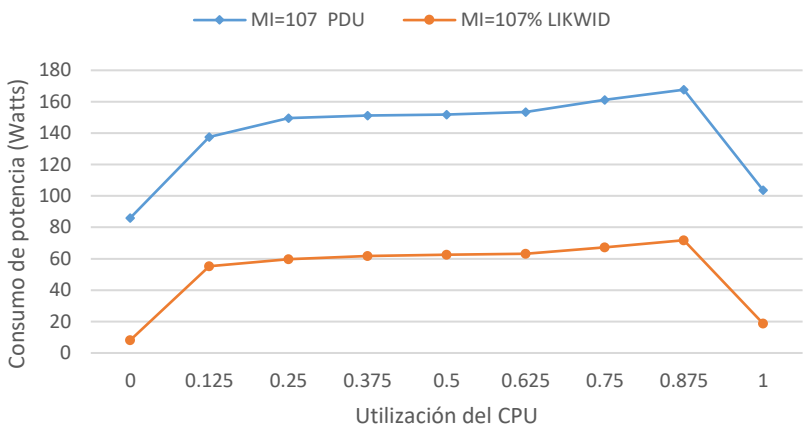
(a) Consumo de energía de MI=100%



(b) Consumo de potencia de MI=100%



(c) Consumo de energía de MI=107%



(d) Consumo de potencia de MI=107%

Figura 13. Perfiles de MI

Los resultados experimentales, muestran que antes de que el sistema alcance una contención significativa que pueda afectar la energía y el rendimiento del sistema, el "makespan" se incrementa ligeramente, mientras la utilización de la memoria del servidor aumenta. Sin embargo, con la utilización de memoria completa (MI = 100% o MI = 107%) se produce el efecto de contención, la degradación del rendimiento aumenta el tiempo de ejecución del sistema hasta 175 segundos para MI = 100% y 1,211 segundos para MI = 107% como se muestra en Figura 14.

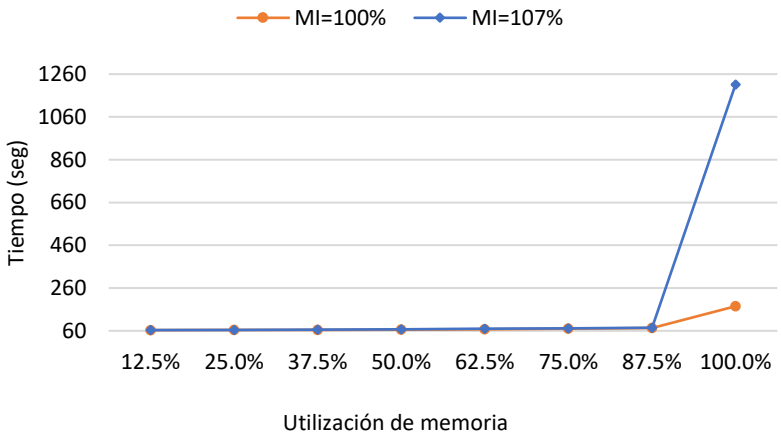
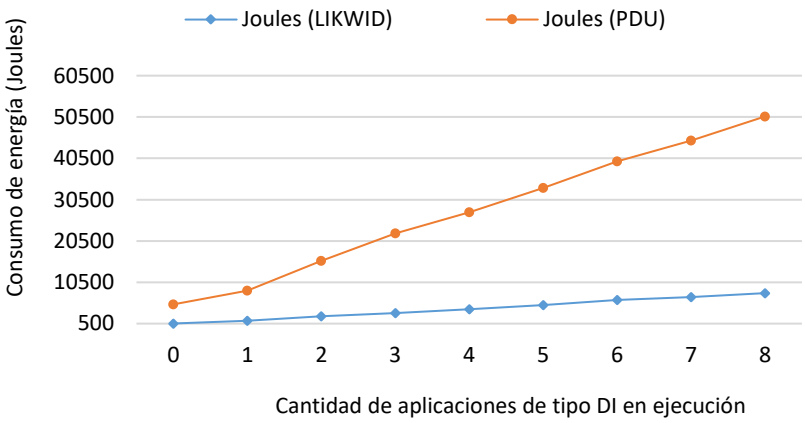


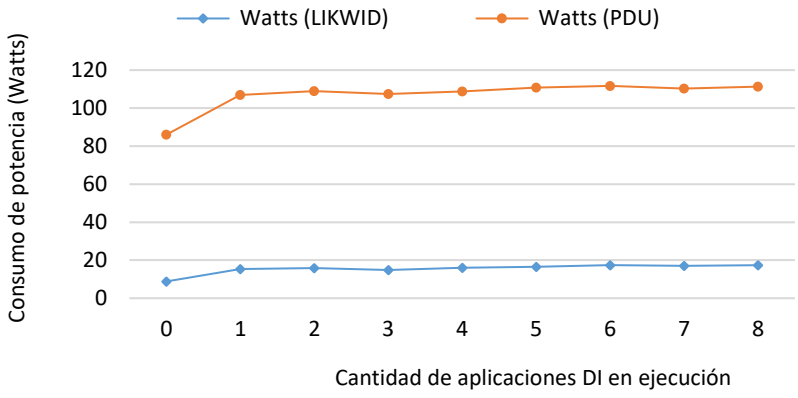
Figura 14. Tiempo de ejecución de MI versus utilización de la memoria

Para aplicaciones de tipo DI, no hubo un impacto significativo en la utilización del CPU. Cuando se aumenta la cantidad de *benchmarks* de tipo DI que se ejecutan simultáneamente en un mismo procesador, la utilización promedio del CPU es del 12%. Sin embargo, como se puede observar en la Figura 15c, hay un incremento lineal en el tiempo de ejecución. El consumo de potencia en el servidor, sigue siendo similar aun cuando se aumenten de una a ocho, la cantidad de aplicaciones de DI, como se muestra en la Figura 15a y la Figura 15b.

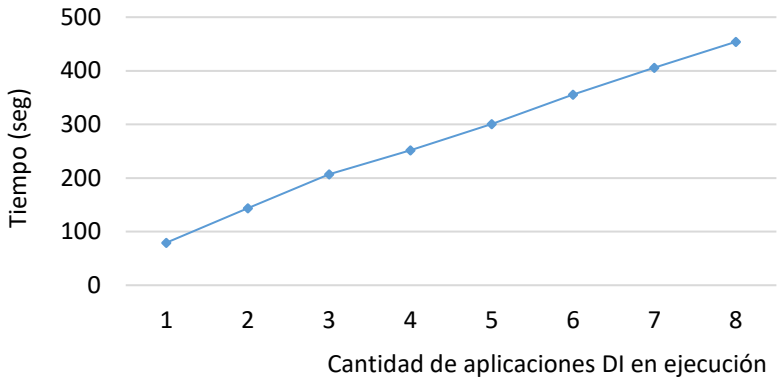
En la Figura 15b, podemos notar que las mediciones de consumo de potencia del PDU se encuentran en el rango de 106W y 111W y las medidas de LIKWID se encuentran entre 15W y 17W.



a) Consumo de energía de DI



b) Consumo de potencia para DI



c) Tiempo de ejecución para DI

Figura 15. Perfiles de la aplicación DI

3.3.2 Perfil energético al combinar los tipos de aplicaciones

En esta sección, realizamos la combinación de aplicaciones para calcular el coeficiente $\boldsymbol{g}(\boldsymbol{\alpha_{CI}})$. Se utiliza dos procesadores homogéneos P_1 y P_2 al 100% de utilización del CPU. Primero, medimos el consumo de energía en ambos procesadores cuando ejecutan sólo un tipo de aplicación. P_1 ejecuta aplicaciones de MI y P_2 ejecuta aplicaciones de CI. Al inicio de la prueba, la concentración de CI en P_1 es cero y uno en P_2. Posteriormente, medimos el consumo de energía cuando intercambiamos aplicaciones MI y CI de una a ocho en ambos procesadores. Al final, la concentración de CI en P_1 es uno y cero en P_2.

La Figura 16 muestra el coeficiente $g(\alpha_{CI})$ que calcula la reducción del consumo de energía versus concentración. El valor del coeficiente más bajo se obtiene cuando la concentración de CI e MI en cada procesador es 0.5; cuatro CI y cuatro aplicaciones MI en cada procesador. Cuando se realiza la combinación de aplicaciones, se obtiene una reducción en el consumo de energía en comparación cuando no se realiza combinación alguna. Cuando la concentración de P_1 es cero y una en P_2 o viceversa, no hay reducción en el consumo de energía, por lo tanto, $g(\alpha_{CI}) = 1$.

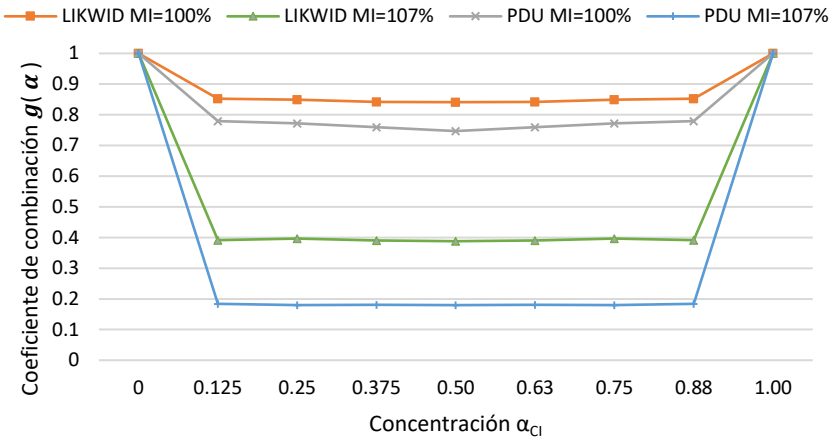


Figura 16. Fracción de consumo de energía de P_1 y P_2

3.3.3 Ecuaciones basadas en perfiles de energía

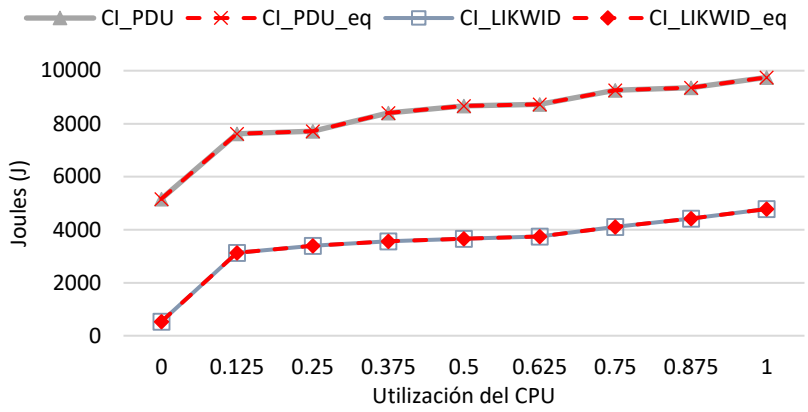
Con base en los perfiles energéticos, para obtener una ecuación para cada tipo de trabajo, que estime el consumo de energía a diferentes niveles de utilización, empleamos interpolación polinómica de Lagrange (Chapra y Canale, 2015).

La Tabla 4, presenta las ecuaciones de consumo de energía a nivel del procesador en función de las mediciones de LIKWID y a nivel del servidor en función de las mediciones del PDU. La columna "experimento" relaciona cada ecuación con cada experimento presentado en la Sección 5.2.

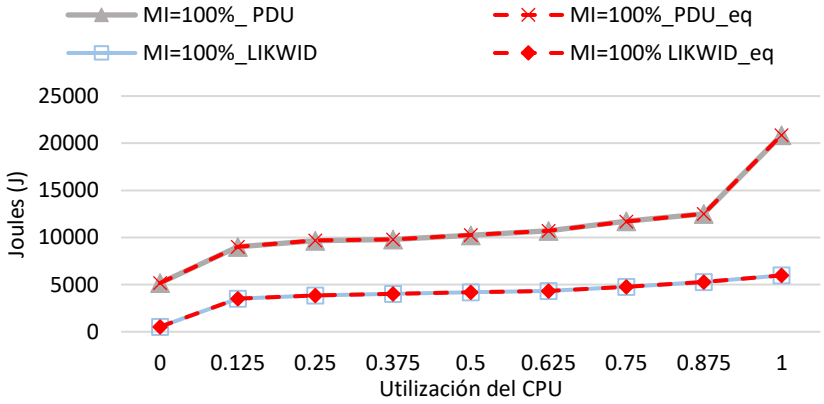
Tabla 4 Ecuación para estimar la energía

Tipo	Experimento	LIKWID	PDU	Ecuación
CI	A, C	•		$F_{CI}(u) = 31.6\,u^8 - 90.4\,u^7 + 71.9\,u^6 + 25.5\,u^5 - 73.1\,u^4 + 46.4\,u^3 - 13.8\,u^2 + 2.07\,u$
CI	B, D		•	$F_{CI}(u) = 5.9\,u^8 - 5.2\,u^7 - 28.1\,u^6 + 63.3\,u - 54.8\,u^4 + 23.5\,u^3 - 5.06\,u^2 + 0.5\,u$
MI= 100%	A	•		$F_{MI}(u) = 95.3\,u^8 - 347.3\,u^7 + 495.1\,u^6 - 339.5\,u^5 + 101.9\,u^4 + 1.2\,u^3 - 8.5\,u^2 + 1.9\,u$
MI= 100%	C		•	$F_{MI}(u) = 79.4\,u^8 - 301.2\,u^7 + 467.1\,u^6 - 379.2\,u^5 + 170.3\,u^4 - 40.1\,u^3 + 3.7\,u^2 + 0.15\,u$
MI= 107%	B	•		$F_{MI}(u) = 406.8\,u^8 - 1449.03\,u^7 + 2099.5\,u^6 - 1579.6\,u^5 + 646.9\,u^4 - 133.4\,u^3 + 8.4\,u^2 + 1.1\,u$
MI= 107%	D		•	$F_{MI}(u) = 426.4\,u^8 - 1515.8\,u^7 + 2213\,u^6 - 1707.6\,u^5 + 743.8\,u^4 - 179.2\,u^3 + 21.03\,u^2 - 0.6\,u$

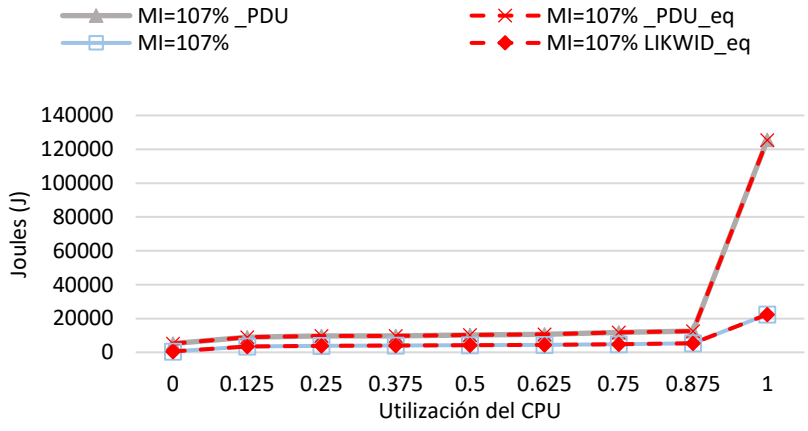
En la Figura 17, comparamos los valores de energía obtenidos en las mediciones empíricas (líneas continuas) con los valores estimados obtenidos con las ecuaciones (líneas discontinuas) para las aplicaciones CI, MI = 100% y MI = 107%.



a) Aplicaciones de CI



b) Aplicaciones de MI100%



c) Aplicaciones de MI 107%

Figura 17. Valor empírico vs. valor estimado

Capítulo 4. Estrategias de asignación con consciencia energética

4.1 Estrategias de asignación

En esta sección, definimos nuestro enfoque de calendarización y diseñamos algoritmos conscientes de la energía, conscientes de la utilización y del tipo de trabajo. Luego, comparamos los algoritmos con estrategias de asignación clásicas. El análisis es similar a nuestro trabajo presentado anteriormente en (Armenta-Cano et al., 2017) donde utilizamos un modelo de energía basado en la medición de energía presentada en la Sección 3.1.

Nuestro enfoque aborda una calendarización de dos niveles (Armenta-Cano et al., 2017; Dorronsoro et al., 2014; Tchernykh et al., 2006, 2010; Tchernykh, Lozano, et al., 2014). En el nivel superior, el sistema, que tiene una vista general de las solicitudes de trabajo y los recursos disponibles, asigna trabajos a máquinas disponibles (admisibles) utilizando un criterio de selección dado. El sistema de manejo de recursos local ejecuta tareas en el nivel inferior.

Proponemos siete estrategias de asignación y las comparamos con tres estrategias de asignación clásicas, ver Tabla 5. Las categorizamos en cuatro grupos según el tipo y la cantidad de información utilizada para la decisión de asignación.

(1) S*in conocimientos*, estrategias clásicas $Rand$ (Aleatorio), $FFit$ (First Fit), RR (Round Robin) (Ramírez-Alcaraz y otros, 2011; Rodríguez Díaz y otros, 2003; Tchernykh, Pecero, et al. ., 2014).

(2) *Consciente de la energía*, estrategia $Mín\_e$ (energía mínima), asigna el trabajo a la máquina que consume la menor cantidad de energía en el instante t.

(3) *Consciente de la utilización*, estrategias Min_u (utilización mínima) y Max_u (utilización máxima) asignan el trabajo a la máquina con la menor o mayor utilización en el instante t respectivamente. Estas estrategias se utilizan para analizar el consumo de energía cuando se minimiza o maximiza la utilización del CPU.

(4) *Consciente del tipo de trabajo*, estrategia Min_ujt (utilización mínima del tipo de trabajo), asigna el trabajo a la máquina con la utilización de trabajos del mismo tipo más baja. Esta estrategia se realiza con el objetivo de equilibrar la utilización de todos los tipos de trabajo. Min_c (concentración mínima) asigna el trabajo a la máquina con la menor concentración de trabajos del mismo tipo. Esta estrategia equilibra las concentraciones de tipos de trabajo sin considerar la utilización. $MinU\_MinC$ (utilización mínima y

concentración mínima) y $MaxU\_MinC$ (utilización máxima y concentración mínima), seleccionan el subconjunto de máquinas admisibles con la mínima utilización total y la máxima utilización total respectivamente. Posteriormente, ambas estrategias utilizan Min_c sobre este subconjunto de máquinas. Estas estrategias equilibran la concentración de trabajos y maximizan o minimizan la utilización.

Tabla 5. Estrategias de calendarización

Tipo	Estrategia	Descripción
Sin conocimiento	$Rand$	Asigna aleatoriamente el trabajo j a una máquina admisible usando una distribución uniforme en el rango $[1..m]$.
	$FFit$ (First Fit)	Asigna el trabajo j a la primer máquina admisible
	RR (Round Robin)	Asigna el trabajo j a una máquina admisible utilizando la estrategia clásica Round Robin
Consciente de la energía	Min_e (Energía mínimo)	Asigna el trabajo j a la máquina admisible con el mínimo consumo de energía $min_{i=1..m}\left(e_i^{proc}(r_j)\right)$ en el instante de tiempo r_j
Consciente de la utilización	Min_u (Utilización mínimo)	Asigna el trabajo j a la máquina admisible con el menor nivel de utilización $min_{i=1..m}(u_i^{proc})$ en el instante de tiempo r_j
	Max_u (Utilización máxima)	Asigna el trabajo j a la máquina admisible con el mayor nivel de utilización $max_{i=1..m}(u_i^{proc})$ en el instante de tiempo r_j
Consciente del tipo de trabajo	$MaxU\_MinC$ (Utilización máxima y Concentración mínima)	Asigna el trabajo j a la máquina admisible que pertenece al subconjunto de máquinas con el mayor nivel de utilización $max_{i=1..m}(u_i^{proc})$ con el menor nivel de concentración de tareas del mismo tipo en el instante de tiempo r_j
	$MinU\_MinC$ (Utilización mínima y Concentración mínima)	Asigna el trabajo j a la máquina admisible que pertenece al subconjunto de máquinas con el menor nivel de utilización $min_{i=1..m}(u_i^{proc})$ con el menor nivel de concentración de tareas del mismo tipo en el instante de tiempo r_j
	Min_ujt (Utilización del tipo de trabajo mínima)	Asigna el trabajo j a la máquina admisible con el menor nivel de utilización de tareas del mismo tipo en el instante de tiempo r_j
	Min_c (Concentración mínima)	Asigna el trabajo j a la máquina admisible con el menor nivel de concentración de tareas del mismo tipo en el instante de tiempo r_j

4.2 Configuración experimental

A continuación, presentamos la configuración experimental, la carga de trabajo, los escenarios de evaluación, y describimos la metodología utilizada para el análisis.

Todos los experimentos se realizan con el simulador CloudSim (Calheiros et al., 2009). CloudSim es un sistema para el modelado y la simulación de infraestructuras y servicios de computación en la nube. Es un simulador estándar que se utiliza para estudiar problemas de gestión de recursos en la nube. Hemos extendido CloudSim para simular un ambiente en línea e incluir nuestros algoritmos usando el lenguaje de programación Java (JDK 7u51).

4.2.1. Carga de trabajo

La evaluación del rendimiento de las estrategias de calendarización, requiere una cuidadosa experimentación en entornos de prueba. Un elemento crítico de estos experimentos, es el uso de cargas de trabajo realistas que puedan representar las condiciones del ambiente que se pretende evaluar. Nuestra carga de trabajo se basa en cargas de HPC reales de “Parallel Workloads Archive” (Feitelson et al., 2014), y de cargas de trabajo de ambientes Grid ("The Grid Workloads Archive," n.d.). Estas cargas de trabajo son adecuadas para nuestros escenarios, donde las máquinas están destinadas a ejecutar trabajos que tradicionalmente se encuentran en ambientes Grids y HPC.

Las cargas de trabajo incluyen nueve archivos: DAS2-Universidad de Ámsterdam, DAS2-Universidad Tecnológica de Delft, DAS2-Universidad de Leiden, DAS2-Universidad de Leiden, KHT, DAS2-Vrije Universidad de Ámsterdam, HPC2N, CTC, y LANL. Se puede encontrar información detallada sobre las características de los registros y las cargas de trabajo en (Feitelson et al., 2014) y ("The Grid Workloads Archive," n.d.).

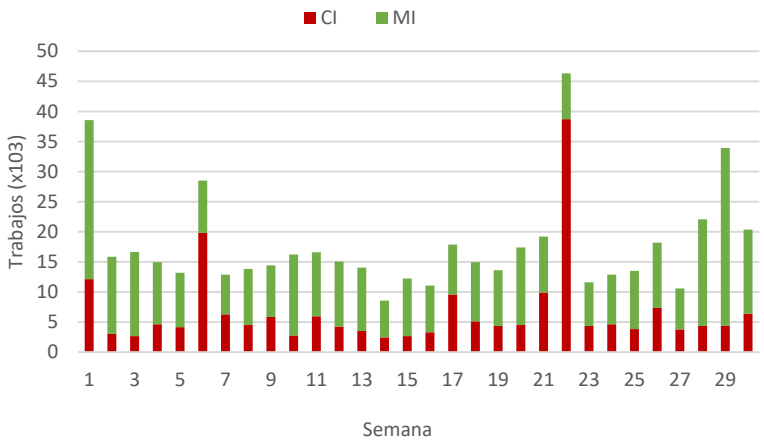
Algunas características de la carga de trabajo, como la distribución de la demanda de trabajo a lo largo del día, el día de la semana y de los husos horarios, pueden afectar a la evaluación de las estrategias. Por lo tanto, utilizamos la normalización del tiempo desplazando las cargas de trabajo por un intervalo de tiempo específico para representar una configuración más realista. Consideramos la normalización de zonas horarias, la normalización del tiempo de grabación y el filtrado de trabajos no válidos. Desplazamos todas las cargas de trabajo al mismo huso horario, por lo que todas las trazas comienzan el mismo día de la semana y a la misma hora del día. La alineación está relacionada con la hora local; por lo tanto, se mantienen las diferencias horarias correspondientes a los husos horarios originales.

Se aplicaron filtros para eliminar trabajos con información inconsistente, por ejemplo, valores negativos del tiempo de envío, el tiempo de ejecución, el número de procesadores asignados, el tiempo solicitado y el ID de usuario; un trabajo con ejecución parcial o fallida y trabajos cancelados, por ejemplo, cancelados

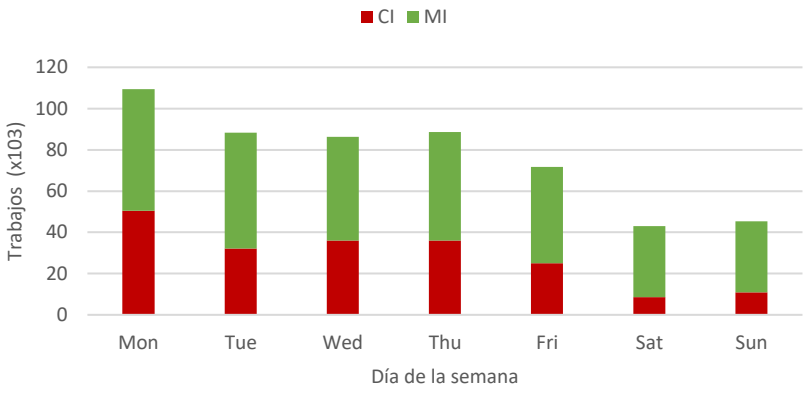
antes de comenzar o durante la ejecución. También añadimos dos campos al Formato estándar de carga de trabajo (swf) para describir el tipo de trabajos y la utilización del CPU.

Nos enfocamos en un problema de calendarización en línea no clarividente, por lo tanto, las decisiones de calendarización deben tomarse sin información completa sobre toda la instancia del problema y cualquier trabajo futuro. Los trabajos llegan uno por uno durante el tiempo. Se desconocen el tiempo de procesamiento de los trabajos tanto al inicio como durante la ejecución. El tiempo de ejecución sólo se conocen cuando los trabajos terminan.

La Figura 18 muestra un histograma con las características de las cargas de trabajo, mostrando la distribución total de trabajos por semana y día. Para obtener valores estadísticos válidos, consideramos una carga de trabajo de 30 semanas. Cada semana contiene el número total de trabajos, el número de trabajos del tipo CI y de tipo MI. Se puede observar que no hay predominio de un tipo de trabajos en los registros. Algunas semanas tienen más trabajos de tipo CI, y otras, más trabajos de tipo MI. Más detalles de la carga de trabajo se presentan en la Figura 19a y Figura 19b. En las cuales se muestra el promedio de trabajos por día de la semana y por hora.

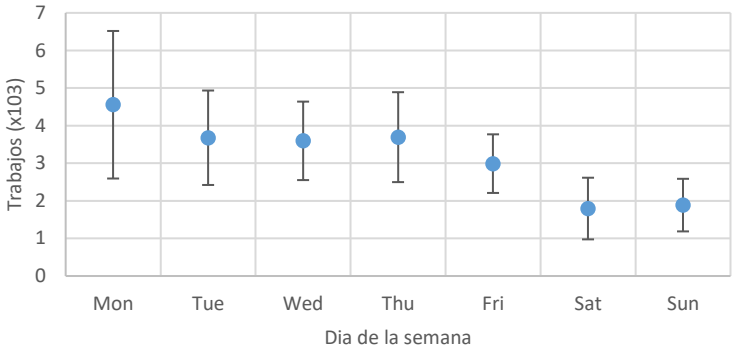


a) Cantidad de trabajos por semana

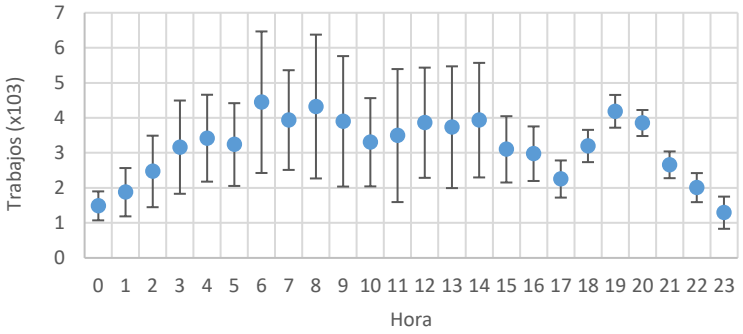


b) Cantidad de trabajos por día de la semana

Figura 18 Distribución del número total de trabajos



c) Promedio de trabajos por día de la semana



d) Promedio de trabajos por hora

Figura 19. Distribución promedio de trabajos.

4.2.2 Sistema de simulación Cloudsim

Para simular el ambiente de la nube e implementar nuestras políticas de asignación en un entorno controlado, donde se puedan reproducir los resultados, es necesaria una plataforma de simulación. Se han desarrollado varios simuladores de Grid, como GridSim (Buyya y Murshed, 2002), SimGrid (Legrand et al., 2003) y GangSim (Dumitrescu y Foster, 2005). Estas herramientas son capaces de modelar y simular las aplicaciones Grid en un entorno distribuido. Sin embargo, ninguno de ellos puede soportar la infraestructura y los requisitos a nivel de aplicación que surgen del paradigma del cómputo en la nube, como el modelado de recursos habilitados para la virtualización bajo demanda.

Una de las principales características que diferencian a una infraestructura de cómputo en Grid de una infraestructura de cómputo en la nube, es el despliegue de herramientas y tecnologías de virtualización. Por lo tanto, a diferencia de los Grids, la nube contiene una capa de virtualización que actúa como entorno de gestión, ejecución y alojamiento para los servicios de aplicaciones. Por otro lado, el simulador CloudSim es una plataforma que se adapta mejor a nuestro problema, ya que es un sistema de simulación moderno dirigido a entornos de cómputo en la nube. Cuenta con la capacidad de soportar el modelado de sistemas y comportamientos de componentes de sistemas de nube tales como centros de datos, máquinas virtuales (VMs) y políticas de suministro de recursos.

CloudSim soporta el modelado y simulación de entornos de cómputo en la nube consistentes tanto en una sola nube como en la federación de nubes (Calheiros et al., 2011).

A continuación, describimos brevemente la plataforma de simulación de CloudSim y mostramos en qué capa se implementaron las estrategias de asignación propuestas dentro de su arquitectura. El diseño multicapa de CloudSim y sus componentes, se observan en la Figura 20. El modelo de CloudSim es presentado en (Calheiros et al., 2011).

La capa de simulación de CloudSim, proporciona soporte para el modelado y simulación de entornos virtualizados de centros de datos basados en la nube, incluyendo interfaces de gestión dedicadas para VMs, memoria, almacenamiento y ancho de banda. Esta capa maneja los problemas fundamentales, como el abastecimiento de hosts a VMs, la gestión de la ejecución de aplicaciones y el monitoreo del estado dinámico del sistema.

Cada componente del servidor también crea un componente del programador de VM. Se puede implementar la política de tiempo compartido o de espacio compartido para asignar núcleos a las VM. En este nivel se implementaron las nuevas estrategias, para el estudio de la eficiencia de diferentes

estrategias que consideran el consumo de energía. Esta implementación puede realizarse ampliando la funcionalidad básica de suministro de VMs. Hay una clara distinción en esta capa relacionada con el abastecimiento de hosts a VMs.

La capa superior en la pila de CloudSim es el Código de Usuario. Esta capa expone la funcionalidad de configuración para servidores (número de máquinas, su especificación), aplicaciones (número de tareas y sus requisitos), VMs (número de usuarios y sus tipos de aplicación), y políticas de calendarización.

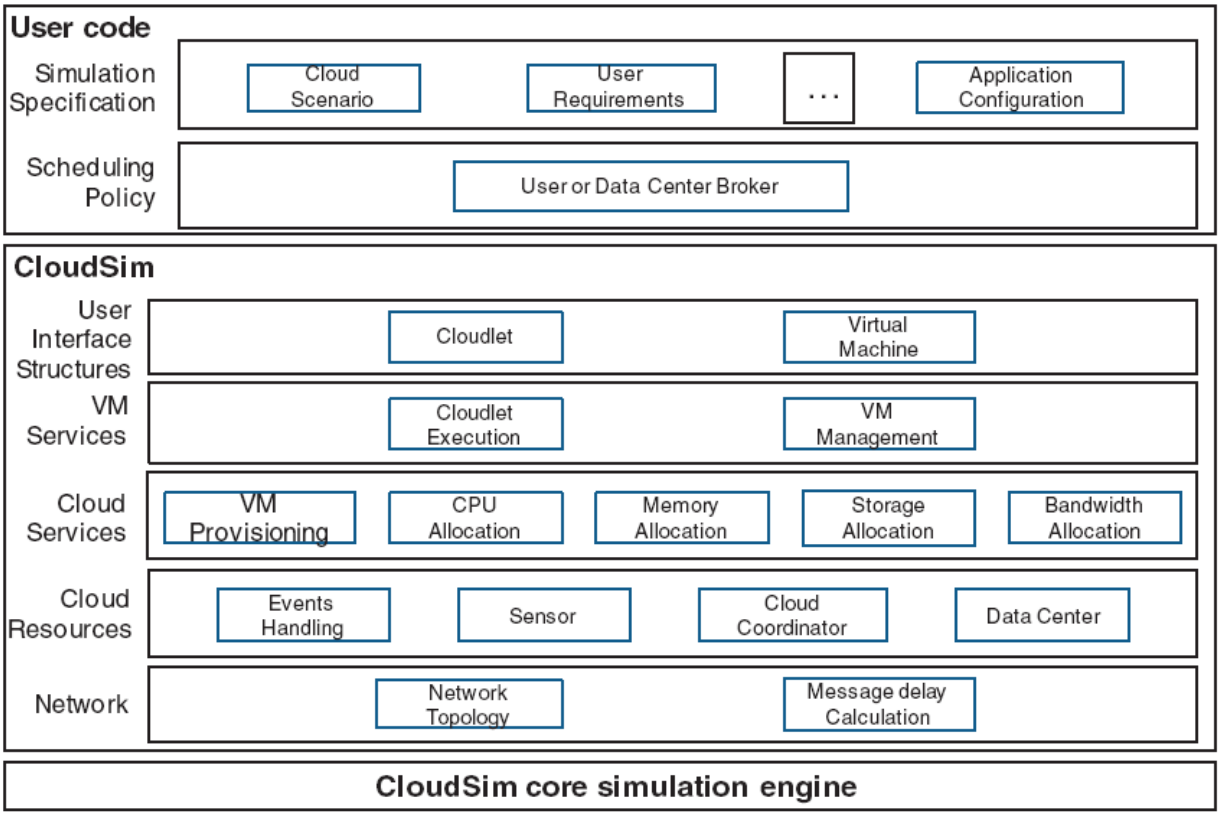


Figura 20. Arquitectura de CloudSim por capas (Calheiros et al. 2011).

Modelado de cargas de trabajo dinámicas

Los desarrolladores de software a menudo desarrollan aplicaciones con un comportamiento dinámico con respecto a los patrones de carga de trabajo, la disponibilidad y los requisitos de escalabilidad. Por lo tanto, un elemento esencial en cualquier entorno de simulación, es el modelado de patrones dinámicos de carga de trabajo impulsados por modelos de aplicaciones o SaaS. Dentro de CloudSim hay una entidad de simulación adicional, que se conoce como “Modelo de Utilización”. Este modelo expone los métodos y variables para definir los recursos y los requisitos a nivel de VM de una aplicación SaaS. CloudSim soporta

el modelado de escenarios de violación de SLA con respecto a parámetros de QoS como disponibilidad, fiabilidad y rendimiento. La falta de metodologías inteligentes para el abastecimiento de máquinas virtuales puede provocar pérdidas de rendimiento en cuanto a tiempo de respuesta, tiempos de espera o fallos en el peor de los casos.

Funcionalidad de CloudSim

Hay diferentes pasos en la simulación de CloudSim. A un nivel abstracto, su funcionamiento puede ser representado por los siguientes siete pasos:

- Paso 1: Inicializar el paquete CloudSim
- Paso 2: Crear los centros de datos
- Paso 3: Crear un broker
- Paso 4: Crear las máquinas virtuales
- Paso 5: Crear las nubes
- Paso 6: Simulación
- Paso 7: Dar salida a los resultados

A continuación, describimos cada uno de los pasos.

En el Paso 1, CloudSim inicializa a los usuarios, establece los planificadores, y se prepara para el siguiente nivel. En el Paso 2, se realiza la creación de los centros de datos. Primero, CloudSim establece las máquinas y asigna el número de núcleos por procesador. Después de esto, establece los costos de procesamiento, almacenamiento y uso del ancho de banda para cada recurso. En el siguiente paso, CloudSim crea el broker, donde cada usuario desarrolla sus políticas asignación de acuerdo a sus objetivos. En esta etapa, implementamos nuestras estrategias de asignación para simular un entorno dinámico.

Las VMs se crean en el Paso 4, los hosts se asignan a una o más VMs basándose en una política de asignación que debe ser definida por el proveedor de servicios de la nube. De forma predeterminada, las VMs se asignan al host según el orden de llegada. CloudSim soporta el abastecimiento de VMs de dos niveles: a nivel de host y a nivel de VM. A nivel de procesador, es posible especificar la potencia de procesamiento de cada núcleo que se asignará a cada VM. A nivel de VM, la VM designa una cantidad fija de procesamiento disponible para los servicios de aplicación individuales (unidades de tarea) alojados por su motor de ejecución. En el Paso 5, se definen las propiedades de los cloudlets (tareas).

La simulación inicia en el paso 6. En este paso, utilizamos nuestro modelo energético para estimar el consumo de energía, que posteriormente, será considerado por nuestras políticas de asignación, y finalmente en el Paso 7, CloudSim imprime los resultados de la simulación.

4.3 Metodología del análisis experimental

4.3.1. Degradación del desempeño

Para proporcionar una guía efectiva para elegir la mejor estrategia, realizamos un análisis del consumo de energía de acuerdo con la metodología de degradación promedio propuesta en Tsafrir et al. (Tsafrir et al., 2007). Esta metodología se aplicó a problemas de calendarización en (Armenta-Cano et al., 2017; Ramírez-Alcaraz et al., 2011; Tchernykh et al., 2016; Tchernykh, Lozano et al., 2014).

Primero, evaluamos la degradación del desempeño (error relativo) de cada estrategia bajo alguna métrica. La evaluación se realiza en relación con la estrategia con el mejor desempeño para la métrica.

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

Promediamos los valores y clasificamos las estrategias. La mejor estrategia con la degradación promedio más baja con respecto al rendimiento tiene el primer puesto. Nótese que tratamos de identificar estrategias que funcionen bien de manera confiable en diferentes escenarios; es decir, tratamos de encontrar un compromiso que considere todos nuestros casos de prueba. El rango de la estrategia puede no ser el mismo para todos los escenarios.

4.3.2. Perfil de desempeño

El enfoque de degradación del desempeño permite analizar los resultados basados en los valores promedio. Sin embargo, una pequeña porción de datos con una desviación significativa en el proceso de evaluación puede influir en las conclusiones. Por ejemplo, tenemos dos casos con diez soluciones. En el primer caso, todas las soluciones son 2 veces peores que la solución óptima. En el caso 2, tenemos nueve soluciones óptimas y una que es once veces peor que la óptima. Si consideramos los valores promedio, ambos casos nos dan la misma solución. Sin embargo, el segundo caso puede ser mejor. Para evitar este

problema, y ayudar con la interpretación de los datos, presentamos perfiles de rendimiento de nuestras estrategias.

El Perfil de desempeño $\delta(\tau)$ es una función no decreciente y fragmentada, que presenta 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 (Eckart, 1999).

La siguiente ecuación muestra cómo calculamos el perfil de desempeño de cada métrica. Donde $resulta$do es la degradación de cada solución encontrada y τ es el factor de degradación que evaluamos en cada caso, para saber qué porcentaje de las soluciones están por debajo de dicho factor. Una vez que tenemos el número de resultados que están por debajo del factor τ evaluado, dividimos entre el número total de experimentos $n_{experiemntos}$

$$\delta_{metrica} = \frac{|resultado : resultado\ \leq \tau|}{n_{experimentos}}$$

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

Capítulo 5. Análisis experimental

5.1 Validación experimental

En esta sección se presentan los resultados experimentales de las estrategias propuestas. Primero, evaluamos todas las estrategias de acuerdo a su degradación de desempeño, usando la metodología descrita en la Sección 4.3.1, y presentamos su clasificación. Posteriormente, brindamos un análisis más detallado basado en los perfiles de desempeño de las estrategias el cual se describe en la Sección 4.3.2.

Llevamos a cabo cuatro experimentos. En los experimentos A y B; utilizamos los datos empíricos recopilados con la herramienta LIKWID. En los experimentos C y D, utilizamos los datos recopilados con el PDU, ver Tabla 4.

En los experimentos A y C, usamos MI = 100%, cada aplicación MI con una demanda del 12.5% de la memoria total del servidor. El tiempo de ejecución en estos experimentos, es casi tres veces mayor que el tiempo del usuario, ver Figura 14.

En los experimentos B y D, usamos MI = 107%, cada aplicación de MI con una demanda del 13.4% de la memoria total del servidor. El tiempo de ejecución de estos experimentos, es casi siete veces mayor en comparación con los observados en los experimentos A y C, ver Figura 14.

5.1.1. Análisis de degradación considerando el consumo de energía

Las Tablas 6, 7, 8 y 9 muestran la degradación del consumo de energía por semana de los cuatro experimentos. Las Degradaciones bajas representan mejores resultados. En los resultados experimentales, observamos que las degradaciones varían significativamente semana tras semana, y de una estrategia a otra.

En el experimento A, las estrategias que maximizan la utilización del CPU presentan mejores resultados. Las estrategias Max_u, $FFit$, y $MaxU\_MinC$, obtienen una degradación del consumo de energía inferior al 1%. La peor estrategia es Min_u con una degradación promedio del 18.3%. Las estrategias clásicas RR y $Rand$ obtienen una degradación promedio de 8.9% y 8.5%, respectivamente. Las estrategias que minimizan la utilización presentan el peor comportamiento *Min_u, Min_e, MinU_MinC* y *Min_ujt*.

Tabla 6 Degradación del consumo de energía por semana para el experimento A (LIKWID)

Semana	FFit	Max_u	MaxU_MinC	Min_e	Min_c	Min_ujt	Min_u	MinU_MinC	Rand	RR
1	0.13%	0.00%	2.27%	26.98%	15.27%	24.55%	29.93%	23.25%	11.18%	14.15%
2	0.00%	1.27%	0.37%	18.43%	12.70%	16.15%	22.10%	17.12%	13.47%	9.51%
3	0.46%	0.23%	0.00%	12.69%	12.12%	15.33%	14.49%	14.34%	8.53%	6.61%
4	0.53%	0.08%	0.00%	13.04%	9.75%	16.09%	14.44%	15.56%	7.55%	7.27%
5	1.04%	0.00%	1.34%	17.10%	14.59%	20.43%	20.81%	20.51%	7.33%	5.95%
6	0.00%	0.28%	0.67%	14.37%	10.71%	19.62%	19.81%	17.96%	7.77%	8.72%
7	0.00%	1.28%	0.86%	21.49%	20.70%	24.39%	23.45%	25.60%	8.09%	17.66%
8	0.00%	0.60%	0.56%	11.41%	9.51%	12.08%	14.78%	13.11%	7.09%	4.52%
9	0.00%	0.22%	2.08%	8.11%	6.13%	9.88%	7.57%	9.79%	5.87%	4.55%
10	1.90%	0.00%	2.41%	14.90%	16.82%	17.91%	18.43%	17.60%	9.19%	12.23%
11	0.16%	0.00%	1.33%	17.95%	10.30%	15.81%	20.10%	19.12%	7.85%	7.42%
12	1.25%	1.22%	0.00%	16.50%	13.57%	16.76%	18.82%	19.93%	9.23%	11.90%
13	0.79%	0.00%	0.52%	21.43%	14.77%	21.26%	24.75%	23.96%	7.55%	12.23%
14	0.00%	0.03%	0.81%	7.82%	5.69%	7.14%	8.16%	6.35%	3.14%	3.23%
15	0.00%	0.88%	0.51%	8.75%	6.36%	7.50%	9.41%	8.18%	5.16%	5.21%
16	0.79%	0.00%	0.38%	16.89%	11.64%	19.19%	18.63%	15.44%	7.92%	0.33%
17	0.52%	0.00%	2.31%	16.35%	4.70%	18.46%	22.99%	13.42%	9.84%	7.84%
18	0.51%	0.00%	1.27%	9.60%	8.22%	11.61%	9.99%	9.02%	5.47%	4.02%
19	0.00%	1.40%	0.57%	16.82%	12.17%	14.58%	15.61%	14.29%	6.15%	7.76%
20	0.24%	0.26%	0.00%	20.12%	10.35%	16.41%	19.50%	16.80%	11.35%	12.49%
21	0.00%	0.49%	0.61%	34.69%	25.88%	28.99%	31.89%	30.99%	14.17%	15.10%
22	0.00%	1.89%	2.77%	27.35%	31.46%	36.89%	37.95%	35.43%	21.46%	21.24%
23	0.00%	1.10%	1.23%	13.81%	6.87%	12.89%	13.73%	12.06%	6.28%	5.65%
24	0.02%	0.18%	0.00%	8.13%	7.92%	11.82%	9.28%	11.23%	5.28%	8.52%
25	2.03%	0.00%	0.91%	12.22%	10.68%	11.30%	14.09%	12.65%	8.69%	10.32%
26	0.76%	0.00%	0.15%	9.32%	6.32%	12.20%	14.60%	11.92%	8.05%	6.29%
27	0.28%	0.00%	1.36%	15.35%	9.75%	16.17%	19.94%	16.42%	5.49%	10.77%
28	3.62%	0.00%	0.42%	16.26%	10.89%	16.35%	15.71%	16.60%	9.38%	8.62%
29	0.00%	0.57%	0.94%	18.93%	12.85%	16.68%	18.25%	19.90%	6.83%	7.01%
30	0.00%	1.71%	2.17%	21.44%	14.52%	18.05%	19.44%	20.49%	10.60%	9.36%

La Figura 21 muestra la diferencia de consumo de energía en kWh entre la mejor estrategia Max_u y la peor estrategia Min_u para el experimento A.

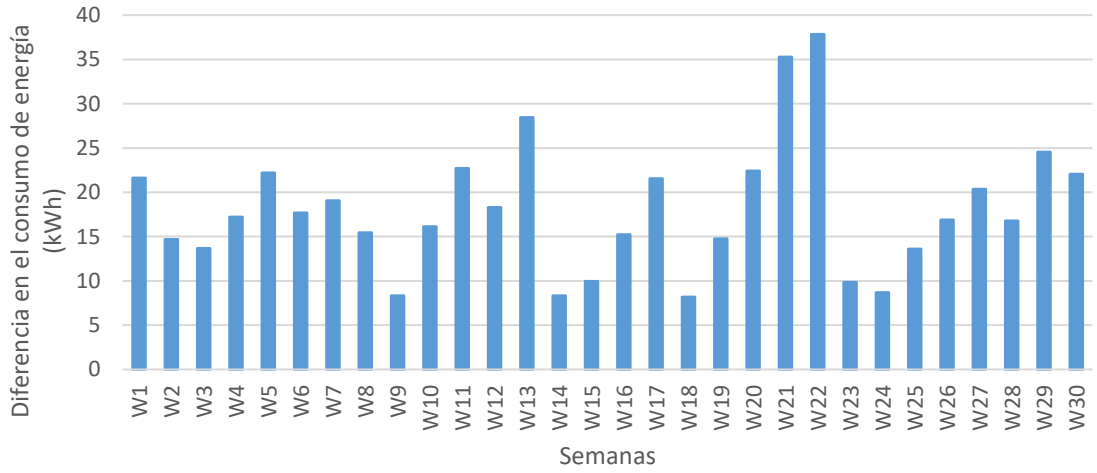


Figura 21. Diferencia de consumo de energía por semana entre la mejor y la peor estrategia (Exp. A).

Los experimentos B y D presentan resultados similares. El valor del coeficiente $g(\alpha_{CI})$ es más bajo en comparación con los experimentos A y B, ver Figura 15. Las estrategias que minimizan la utilización presentan el peor comportamiento $Min\_u, Min\_e, MinU\_MinC\ y\ Min\_ujt$. Por otro lado, las estrategias $FFit$ y Max_u que maximizan la utilización, presentan los mejores resultados. La estrategia $MaxU\_MinC$, maximiza la utilización y realiza un balance en la concentración de tareas. Esta estrategia obtiene el mejor resultado en casi todas las semanas.

En el experimento B, $MaxU\_MinC, FFit$ y Max_u son las mejores estrategias con una degradación promedio de 0%, 7.8% y 8.1%, respectivamente. Min_u es la peor estrategia con una degradación del 18.2%. Las estrategias clásicas RR y $Rand$ obtienen una degradación promedio de 10.9% y 11.4%, respectivamente.

Tabla 7 Degradación del consumo de energía por semana para el experimento B (LIKWID)

Semana	FFit	Max_u	MaxU_MinC	Min_e	Min_c	Min_ujt	Min_u	MinU_MinC	Rand	RR
1	9.89%	11.96%	0.00%	27.85%	19.73%	24.06%	29.75%	23.10%	14.95%	17.50%
2	6.27%	10.27%	0.00%	19.82%	13.72%	16.92%	22.92%	17.86%	17.75%	12.22%
3	5.25%	5.15%	0.00%	14.97%	11.97%	15.08%	16.58%	14.12%	10.13%	8.99%
4	3.71%	2.85%	0.00%	11.36%	10.43%	16.42%	14.36%	16.22%	9.48%	9.24%
5	5.72%	4.34%	0.00%	14.17%	13.54%	19.23%	19.67%	19.32%	6.87%	5.33%
6	0.00%	0.23%	0.07%	17.62%	10.12%	18.94%	19.05%	17.19%	7.53%	8.07%
7	2.15%	3.78%	0.00%	21.60%	19.45%	23.05%	22.37%	24.67%	12.67%	18.66%
8	2.48%	3.64%	0.00%	13.26%	8.96%	11.37%	14.33%	12.47%	8.02%	4.72%
9	6.10%	8.21%	0.00%	7.48%	4.39%	7.72%	6.14%	8.01%	9.73%	6.63%
10	3.82%	4.29%	0.00%	10.94%	15.07%	15.33%	15.87%	15.05%	8.79%	11.87%
11	6.96%	6.83%	0.00%	15.86%	10.03%	15.35%	19.82%	18.61%	9.31%	8.40%
12	6.43%	9.09%	0.00%	22.24%	13.35%	16.70%	19.54%	20.55%	10.28%	12.27%
13	16.98%	18.68%	0.00%	28.72%	22.79%	25.93%	30.22%	29.34%	18.96%	21.57%
14	16.50%	18.01%	0.00%	11.35%	7.37%	8.70%	11.86%	7.92%	16.36%	14.63%
15	24.36%	23.49%	0.00%	15.51%	9.43%	10.50%	13.62%	11.46%	14.15%	13.50%
16	3.36%	3.29%	0.00%	15.26%	11.03%	18.45%	18.07%	14.83%	8.07%	1.06%
17	1.09%	0.69%	0.00%	15.90%	2.62%	15.73%	20.40%	10.98%	8.53%	6.94%
18	2.67%	5.01%	0.00%	6.98%	6.76%	9.97%	8.62%	7.45%	5.12%	3.39%
19	6.78%	8.65%	0.00%	13.64%	13.10%	15.41%	16.78%	15.44%	9.30%	9.85%
20	3.52%	4.28%	0.00%	16.20%	10.63%	16.48%	20.47%	16.93%	12.93%	13.56%
21	10.30%	7.60%	0.00%	34.80%	26.06%	28.88%	31.97%	30.97%	16.52%	17.84%
22	7.29%	5.51%	0.00%	29.64%	28.07%	33.33%	34.43%	32.05%	21.76%	20.22%
23	3.28%	3.97%	0.00%	10.46%	5.98%	11.33%	12.27%	10.52%	7.81%	5.42%
24	4.90%	3.97%	0.00%	10.52%	7.79%	11.69%	9.27%	10.99%	9.31%	9.43%
25	13.54%	11.15%	0.00%	12.96%	9.58%	10.10%	13.78%	11.44%	10.82%	13.75%
26	15.10%	13.51%	0.00%	12.61%	6.56%	11.77%	14.50%	11.58%	9.45%	8.69%
27	8.00%	6.79%	0.00%	15.12%	8.13%	14.37%	18.96%	14.72%	6.33%	10.51%
28	10.45%	4.34%	0.00%	13.66%	10.34%	15.64%	15.05%	15.89%	11.40%	10.54%
29	5.32%	5.93%	0.00%	19.50%	11.65%	15.29%	17.05%	18.46%	6.95%	7.18%
30	15.33%	21.71%	0.00%	22.32%	16.19%	17.99%	20.52%	19.08%	21.31%	14.08%

Cuando se produce un efecto de contención, las estrategias que maximizan la utilización obtienen los mejores resultados. La Figura 22 presenta la diferencia de consumo de energía del experimento B para cada semana entre las estrategias $MaxU\_MinC$ y Min_u.

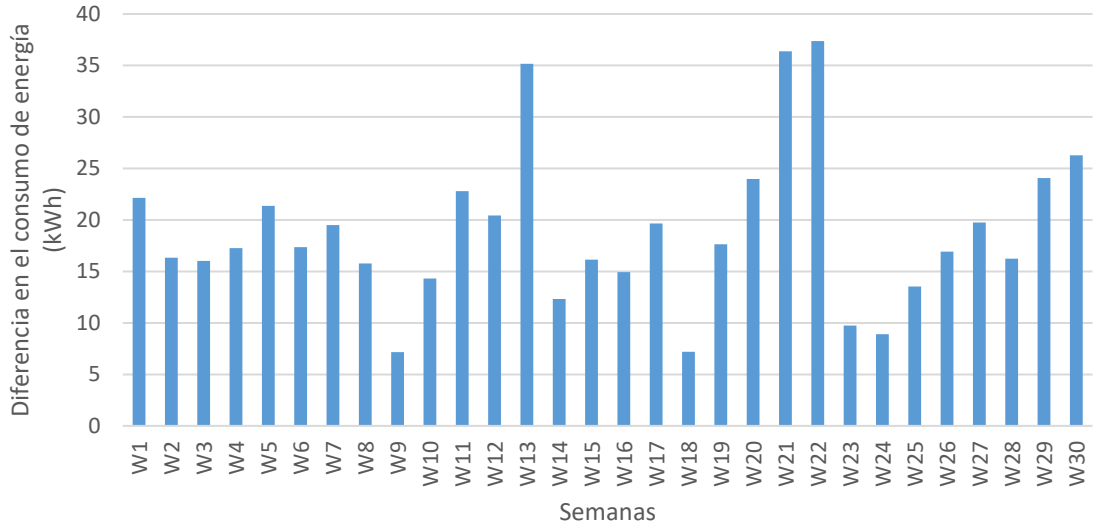


Figura 22. Diferencia de consumo de energía por semana entre la mejor y la peor estrategia (Exp. B).

La Figura 23 presenta la diferencia de consumo de energía entre las estrategias $MaxU\_MinC$ y Min_u en kW/h para el experimento C.

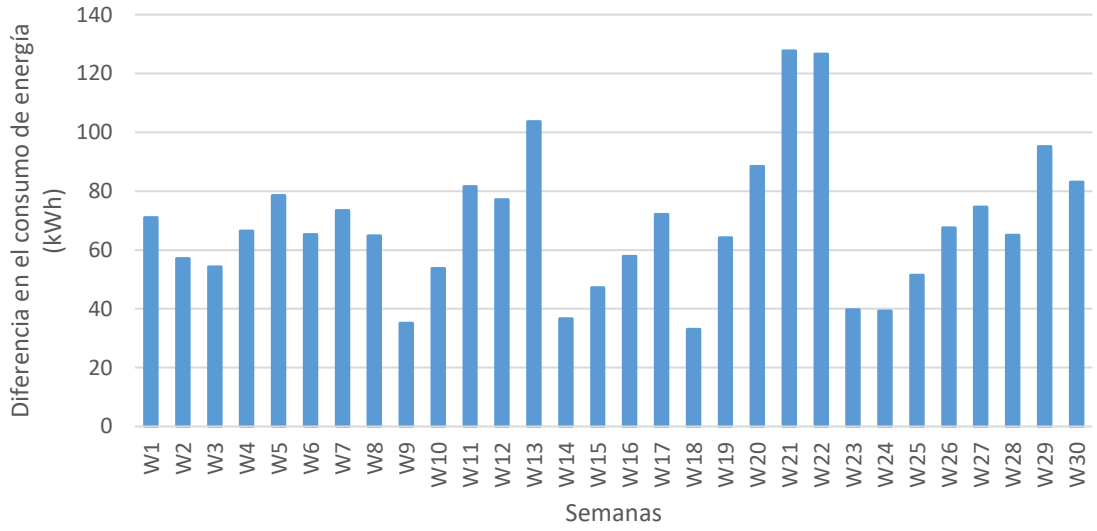


Figura 23. Diferencia de consumo de energía por semana entre la mejor y la peor estrategia (Exp. C).

En el experimento C, las estrategias $MaxU\_MinC$, Max_u y $FFit$ obtienen los mejores resultados en todas las semanas; donde $MaxU\_MinC$ que considera el tipo de aplicaciones supera a las otras dos. Por otro lado, Min_u obtiene la peor degradación.

Tabla 8. Degradación del consumo de energía por semana para el experimento C (PDU)

Semana	FFit	Max_u	MaxU_MinC	Min_e	Min_c	Min_ujt	Min_u	MinU_MinC	Rand	RR
1	0.00%	0.31%	0.32%	31.90%	17.82%	28.79%	34.39%	26.98%	13.09%	16.52%
2	0.87%	2.69%	0.00%	22.97%	16.72%	22.01%	28.70%	23.10%	18.15%	12.99%
3	1.69%	1.06%	0.00%	17.75%	16.00%	20.61%	20.11%	19.52%	11.91%	10.11%
4	1.16%	0.47%	0.00%	16.83%	14.21%	22.42%	20.04%	22.00%	10.96%	10.69%
5	1.71%	0.00%	0.54%	24.51%	18.66%	27.02%	26.95%	26.88%	9.61%	9.73%
6	0.07%	0.00%	0.42%	21.28%	15.19%	26.69%	26.21%	24.71%	11.10%	11.96%
7	0.00%	1.31%	0.15%	24.55%	25.89%	32.07%	30.56%	33.41%	12.18%	23.04%
8	0.05%	0.71%	0.00%	14.25%	12.92%	17.97%	21.36%	19.20%	10.55%	7.06%
9	0.21%	0.00%	0.44%	11.45%	8.04%	13.76%	11.37%	13.91%	8.17%	6.69%
10	1.98%	0.00%	1.34%	18.07%	20.13%	22.58%	23.17%	22.22%	11.38%	15.33%
11	0.48%	0.11%	0.00%	16.61%	12.85%	20.72%	25.67%	24.67%	10.15%	9.83%
12	2.89%	2.96%	0.00%	24.14%	19.04%	24.18%	26.88%	28.24%	13.31%	16.81%
13	3.54%	2.70%	0.00%	28.92%	20.38%	28.19%	32.00%	31.47%	12.09%	17.55%
14	2.70%	2.47%	0.00%	12.91%	7.97%	10.55%	12.73%	9.86%	7.06%	6.77%
15	4.23%	4.48%	0.00%	13.30%	9.49%	11.42%	14.36%	12.59%	8.89%	8.70%
16	1.47%	0.23%	0.00%	21.42%	15.74%	26.12%	25.34%	21.69%	11.09%	4.47%
17	0.80%	0.00%	2.06%	23.17%	7.57%	24.91%	29.49%	19.44%	13.60%	11.15%
18	0.38%	0.00%	0.20%	11.33%	10.84%	16.21%	14.51%	13.25%	7.43%	5.86%
19	1.09%	2.44%	0.00%	17.56%	16.46%	20.46%	22.14%	20.53%	9.32%	11.48%
20	1.05%	0.73%	0.00%	23.04%	14.75%	23.39%	27.30%	24.17%	16.03%	17.44%
21	1.34%	1.21%	0.00%	40.10%	32.85%	37.96%	40.62%	40.29%	18.80%	19.91%
22	0.00%	1.08%	1.10%	39.09%	35.87%	42.85%	43.41%	41.01%	25.06%	24.76%
23	0.00%	0.78%	0.34%	16.01%	8.86%	17.09%	18.41%	16.35%	8.70%	7.52%
24	1.19%	1.07%	0.00%	15.48%	11.56%	17.52%	14.88%	17.06%	9.08%	12.52%
25	3.79%	1.08%	0.00%	16.94%	13.46%	14.94%	18.89%	16.84%	11.59%	13.95%
26	3.59%	2.30%	0.00%	15.22%	9.85%	18.07%	21.14%	17.83%	12.00%	10.10%
27	0.99%	0.00%	0.20%	17.22%	12.78%	21.52%	26.14%	22.12%	7.64%	14.01%
28	5.29%	0.25%	0.00%	18.22%	14.68%	22.07%	21.84%	22.50%	13.33%	12.23%
29	0.39%	0.70%	0.00%	26.87%	16.43%	22.39%	24.42%	26.40%	9.31%	9.59%
30	1.00%	3.44%	0.00%	23.07%	17.11%	21.65%	23.47%	24.35%	13.77%	11.53%

Finalmente, en el experimento D, la estrategia $MaxU\_MinC$ logra la menor degradación del rendimiento en todas las semanas, obteniendo los valores más bajos del coeficiente $g(\alpha_{CI})$, ver Figura 16. La Figura 24 presenta la diferencia de consumo de energía para cada semana entre las estrategias $MaxU\_MinC$ y Min_u.

Tabla 9. Degradación del consumo de energía por semana para el experimento D (PDU)

Semana	FFit	Max_u	MaxU_MinC	Min_e	Min_c	Min_ujt	Min_u	MinU_MinC	Rand	RR
1	30.82%	36.79%	0.00%	39.73%	36.01%	34.89%	41.59%	33.82%	29.19%	31.75%
2	18.19%	27.20%	0.00%	30.93%	20.86%	25.65%	32.61%	26.72%	30.77%	21.37%
3	14.25%	13.97%	0.00%	21.51%	16.00%	20.56%	26.13%	19.50%	16.42%	16.64%
4	9.54%	7.80%	0.00%	17.94%	16.17%	23.55%	20.08%	24.02%	16.19%	16.02%
5	17.09%	14.39%	0.00%	24.25%	19.60%	27.86%	27.87%	27.79%	11.76%	11.42%
6	1.10%	0.92%	0.00%	21.19%	14.95%	26.36%	25.70%	24.18%	11.75%	11.55%
7	7.72%	10.04%	0.00%	28.58%	25.64%	31.90%	30.98%	34.27%	26.78%	28.54%
8	8.02%	10.19%	0.00%	14.73%	13.34%	18.23%	22.19%	19.60%	14.68%	9.24%
9	21.20%	25.96%	0.00%	18.81%	9.00%	14.06%	13.21%	15.15%	23.73%	17.53%
10	11.93%	16.13%	0.00%	16.76%	21.44%	21.88%	22.49%	21.53%	15.77%	20.01%
11	21.91%	21.56%	0.00%	19.80%	16.29%	24.12%	29.61%	28.09%	18.00%	16.36%
12	16.71%	23.93%	0.00%	32.03%	18.98%	24.83%	29.42%	30.75%	16.40%	18.19%
13	47.43%	53.04%	0.00%	42.76%	43.50%	42.93%	48.78%	48.08%	43.77%	44.04%
14	48.30%	51.94%	0.00%	25.93%	14.91%	17.35%	25.15%	16.66%	44.23%	39.10%
15	69.79%	65.40%	0.00%	27.98%	19.34%	21.12%	27.31%	23.10%	34.26%	32.18%
16	9.31%	9.97%	0.00%	23.05%	15.69%	26.05%	25.60%	21.87%	12.83%	7.72%
17	6.22%	5.69%	0.00%	20.63%	6.30%	22.65%	27.81%	17.70%	14.54%	13.11%
18	9.21%	16.36%	0.00%	13.18%	10.69%	15.92%	14.66%	12.97%	10.01%	7.64%
19	20.56%	23.20%	0.00%	25.24%	21.01%	24.99%	27.46%	25.91%	19.44%	18.92%
20	9.68%	11.26%	0.00%	28.86%	15.70%	24.00%	30.20%	24.91%	20.35%	20.45%
21	30.13%	21.60%	0.00%	37.46%	35.90%	40.49%	43.74%	43.14%	27.17%	29.34%
22	25.04%	16.65%	0.00%	38.62%	35.36%	42.23%	43.05%	40.77%	33.47%	29.72%
23	11.46%	11.27%	0.00%	14.97%	9.76%	16.65%	18.26%	15.91%	15.96%	10.10%
24	14.21%	11.20%	0.00%	20.47%	11.51%	17.80%	15.30%	16.99%	20.00%	15.27%
25	36.56%	32.74%	0.00%	19.85%	13.62%	14.94%	21.31%	16.85%	20.09%	25.95%
26	42.16%	38.64%	0.00%	16.24%	11.19%	17.99%	21.86%	17.93%	16.43%	17.20%
27	24.72%	21.22%	0.00%	21.12%	12.51%	21.21%	28.13%	22.11%	13.56%	17.33%
28	24.51%	12.78%	0.00%	21.32%	14.75%	22.00%	21.78%	22.44%	20.07%	18.68%
29	16.94%	17.39%	0.00%	27.02%	16.43%	22.33%	24.74%	26.33%	12.52%	12.93%
30	47.03%	61.93%	0.00%	40.11%	28.37%	28.67%	33.57%	27.99%	48.49%	30.38%

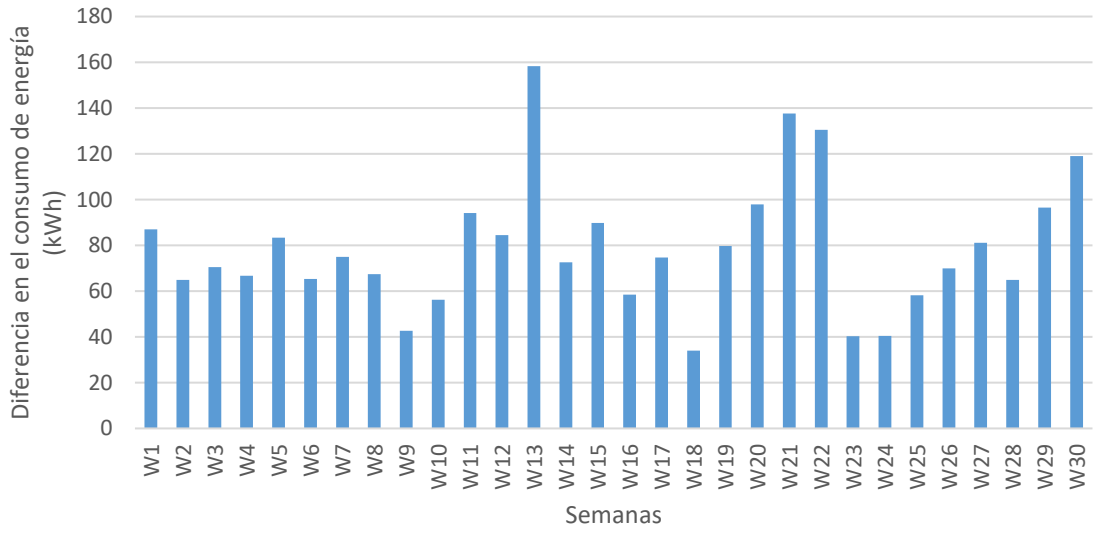


Figura 24. Diferencia de consumo de energía por semana entre la mejor y la peor estrategia (Exp. D)

En la Figura 25 se observa la degradación del consumo de energía promedio de 30 semanas en los experimentos A, B, C y D (véanse las Tablas 6 a 9).

La Figura 25a presenta los resultados de los experimentos A y B a nivel del procesador usando la herramienta LIKWID, y la Figura 25b presenta los resultados de los experimentos C y D a nivel de servidor usando el PDU.

La degradación de energía promedio para los experimentos A y B se muestra en la Figura 25a. La estrategia Min_u obtiene la peor degradación de energía con 18.3%, en promedio. Las estrategias Max_u, $FFit$ y $MaxU\_MinC$ son las mejores estrategias con la degradación de desempeño promedio más baja y tienen los rangos 1, 2 y 3, respectivamente. La diferencia entre Max_u y Min_u es 17.83%. En el mejor de los casos, cuando la concentración $\alpha_{CI} = 0.5$, el valor del coeficiente es $g(\alpha_{CI}) = 0.84$, la energía se reduce en un 16%.

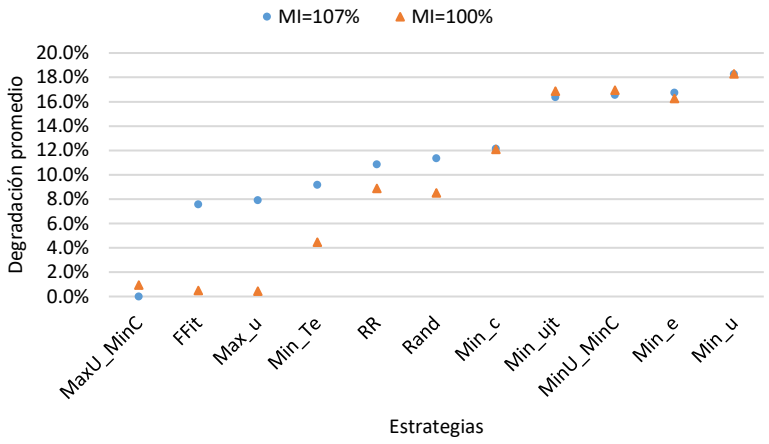
En la Figura 25b, para los experimentos C y D, observamos resultados similares. Las estrategias con las peores degradaciones de energía son *Min_u* y *Min_e*. Las estrategias que consideran el tipo de aplicación presentan mejores resultados. *MaxU_MinC* es la mejor estrategia con la degradación del desempeño más baja.

La principal diferencia entre los experimentos B y D es cuando se produce un efecto de contención. El consumo de energía del disco se considera entre otros elementos y presenta los valores más bajos del coeficiente $g(\alpha_{CI}) = 0.2$.

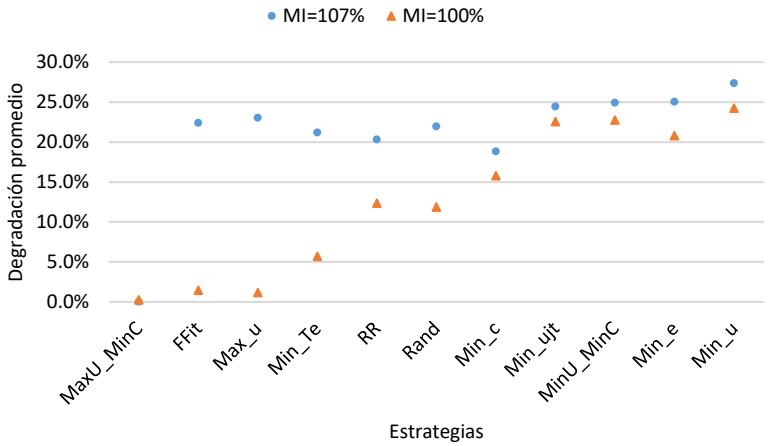
La estrategia *Min_c* obtiene el segundo lugar en el rango de estrategias con una degradación del consumo de energía del 18.9%.

En el experimento C, cuando se produce una combinación de aplicaciones CI y MI, se muestra una reducción en el consumo de energía de casi 40%, ver Figura 16.

En la Figura 25, vemos que las estrategias *MaxU_MinC, Max_u* y *FFit* presentan una reducción del consumo de energía de casi 40% y están clasificadas en 1, 2 y 3, respectivamente.



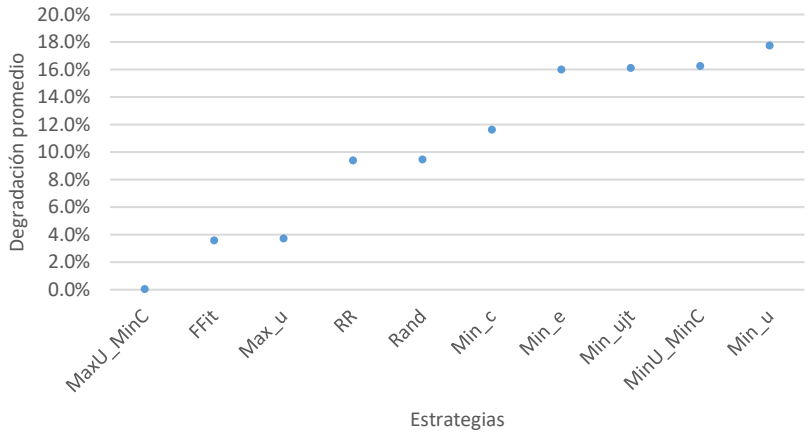
(a) A nivel del procesador (A, B)



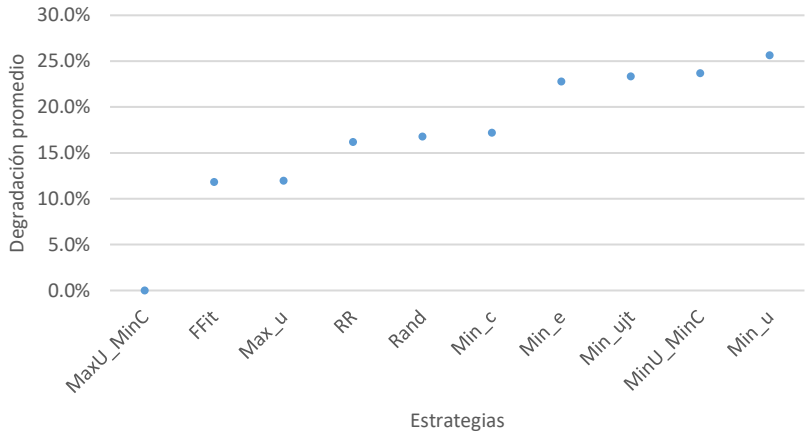
(b) A nivel del servidor (C, D)

Figura 25. Degradación promedio del consumo de energía por estrategia

La Figura 26a presenta los resultados promedio de los experimentos A y B (a nivel de procesador usando la herramienta LIKWID), y la Figura 26b presenta los resultados promedio de los experimentos C y D (a nivel de servidor usando PDU). Podemos observar resultados similares en ambas figuras, la estrategia $MaxU\_MinC$ que maximiza la utilización y distribuye diferentes aplicaciones sobre los recursos físicos, en promedio presenta los mejores resultados en cuanto a consumo de energía.



(a) Degradación promedio del consumo de energía por estrategia a nivel de procesador



(b) Degradación promedio del consumo de energía por estrategia a nivel de servidor

Figura 26. Degradación del consumo promedio de energía por estrategia

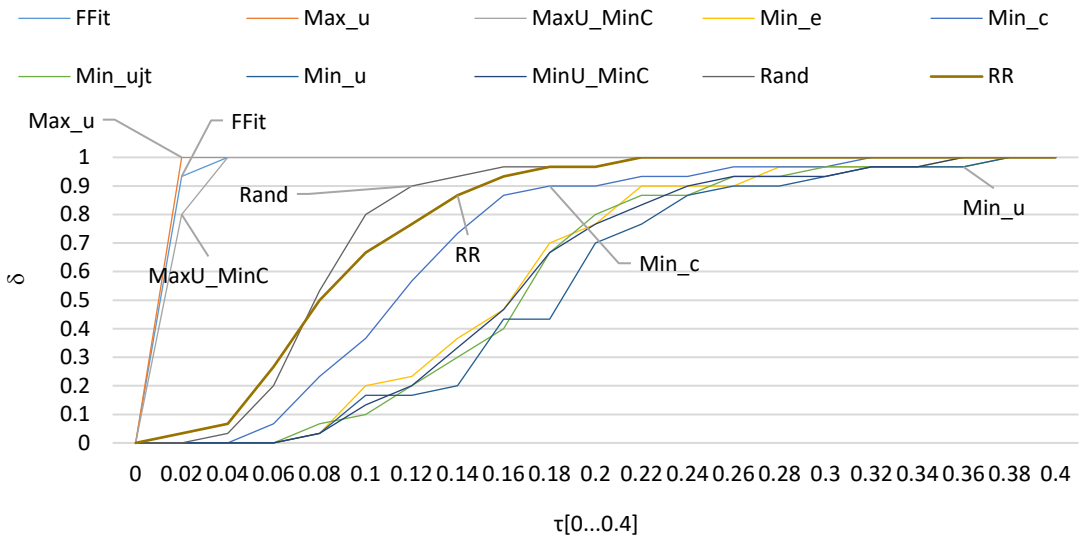
5.1.2. Análisis del perfil de desempeño

Como se menciona en la sección 4.3.2, las conclusiones basadas en valores promedio pueden tener algunos aspectos negativos. Una pequeña porción de los casos del problema con alta desviación podría cambiar el valor promedio y las conclusiones basadas en promedios. Para analizar los resultados con más detalle, presentamos los perfiles de desempeño de las estrategias.

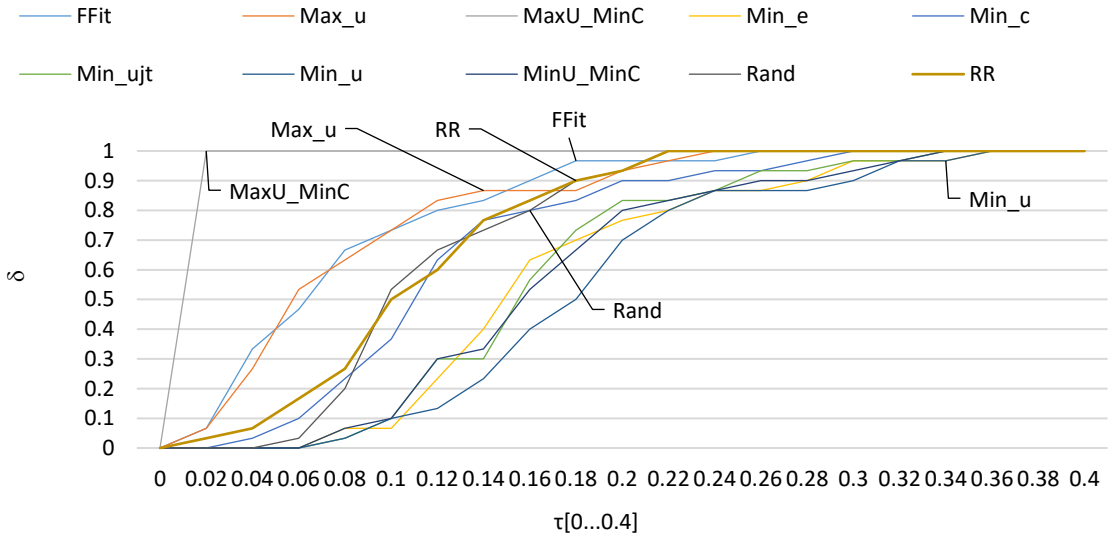
En la Figura 27, presentamos los perfiles de desempeño de las diez estrategias de acuerdo con el consumo de energía en el intervalo $\tau = [0\,, 0.4]$ para los experimentos. La Figura 27a muestra los resultados del

experimento A. Si elegimos un factor de 0.02 del mejor resultado como el alcance de nuestro interés, *Max_u* tiene casi un 100% de probabilidad de ser un ganador.

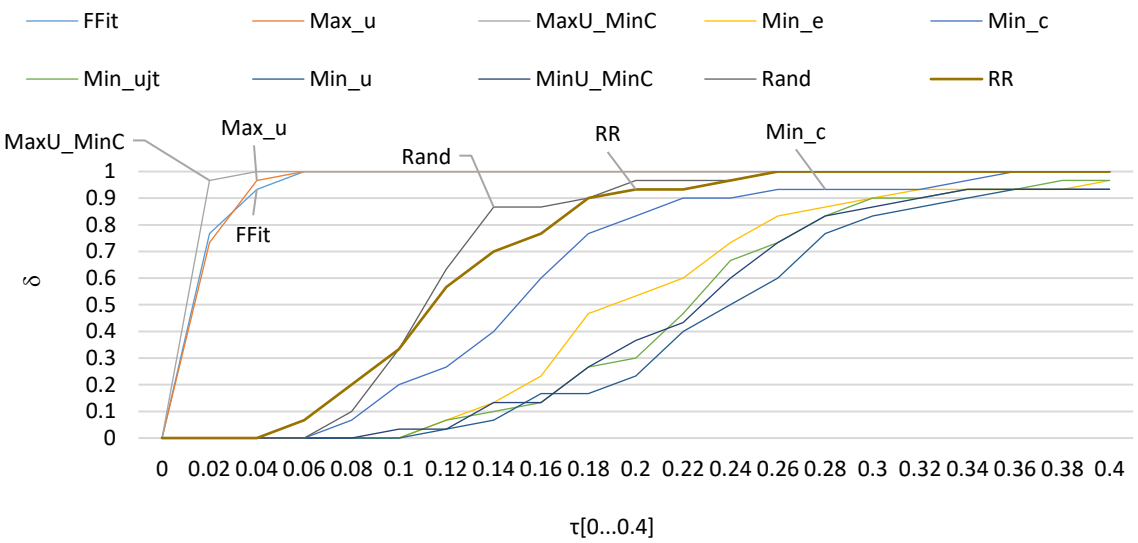
En los experimentos B, C y D (Figura 27b, Figura 27c y Figura 27d); observamos que la estrategia *MaxU_MinC* presenta los mejores resultados. Si elegimos un factor de 0.01 del mejor resultado como el alcance de nuestro interés, el porcentaje de soluciones de *MaxU_MinC* que gana en un problema determinado es del 100%.



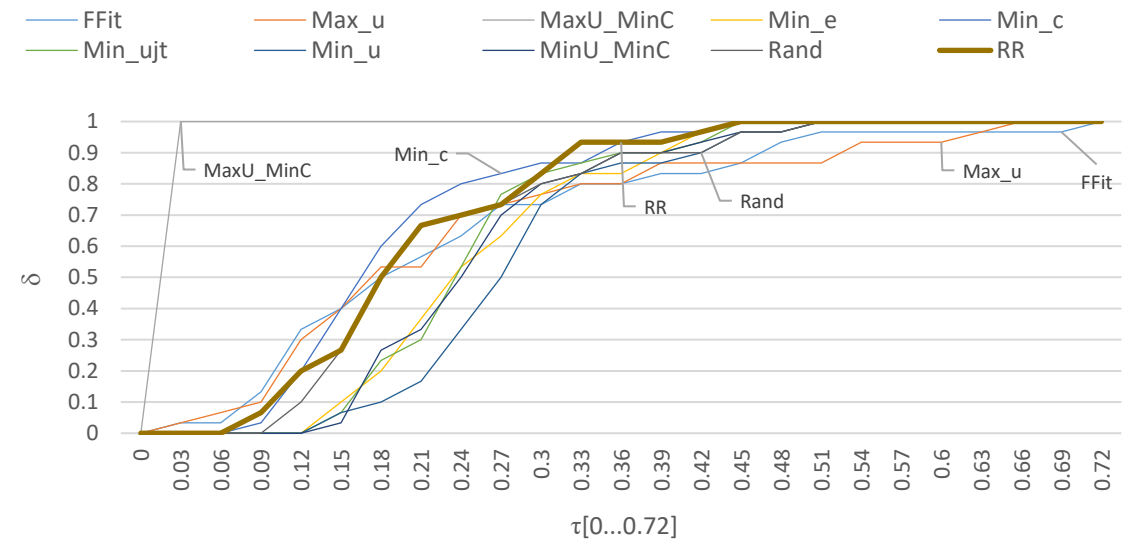
(a) Experimento A



(b) Experimento B



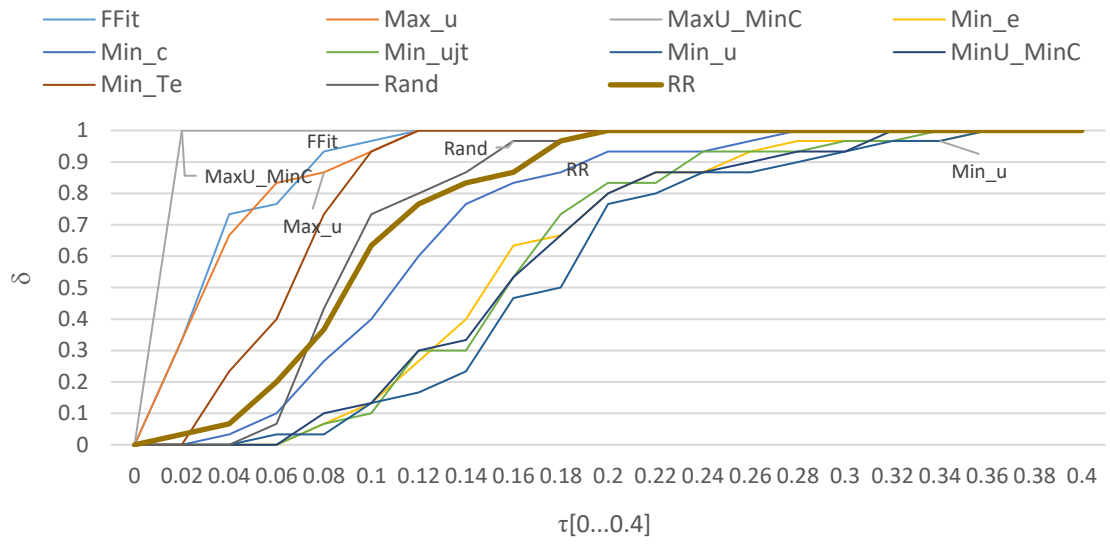
(c) Experimento C



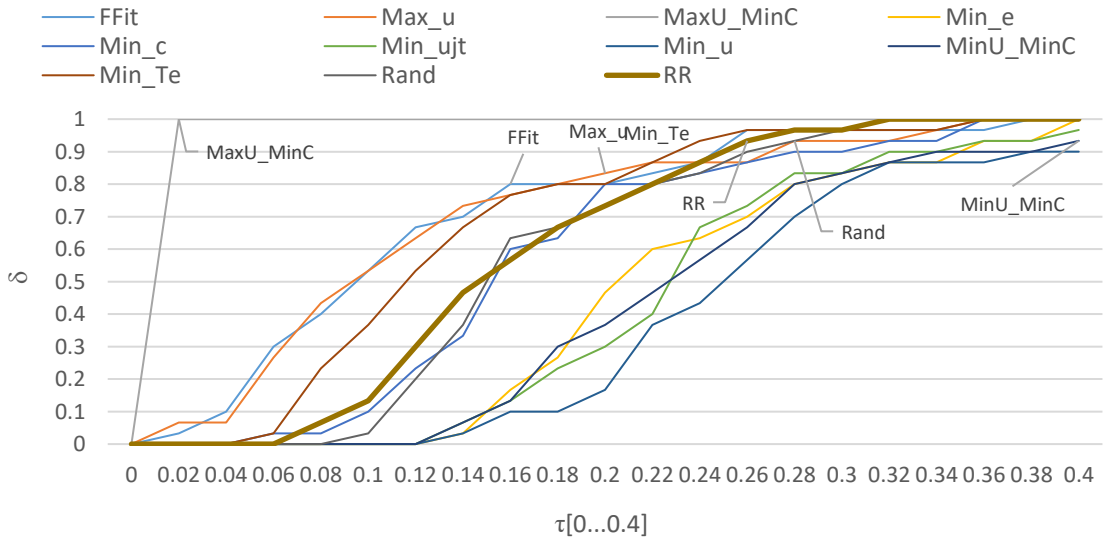
(d) Experimento D

Figura 27. Perfil del desempeño de las 10 estrategias.

La Figura 28 muestra los resultados a nivel del procesador y a nivel del servidor. La Figura 28a y la Figura 28b presentan resultados similares. La estrategia *MaxU_MinC* tiene la clasificación más alta y la mayor probabilidad de ser la mejor estrategia. *Max_u* y *FFit* también tienen una alta probabilidad de obtener un buen resultado.



(a) A nivel del procesador (A, B)



(b)A nivel del servidor (C, D)

Figura 28. Perfil del desempeño promedio de la degradación del consumo de energía de las 10 estrategias.

Para más detalles de los cuatro experimentos, véase el Apéndice.

Capítulo 6. Conclusiones

Las aplicaciones que se ejecutan en la nube pueden estar ligadas al procesamiento, donde requieren mayormente el CPU, a los recursos de E/S, al demandar un gran ancho de banda o almacenamiento o ligadas a la memoria, etc. Cuando las tareas de un tipo se asignan al mismo recurso, se pueden producir conflictos. Dichos conflictos se pueden presentar en: CPU, memoria, disco y red; provocando degradación del rendimiento del sistema y el aumento del consumo de energía.

La principal contribución de este trabajo se divide en dos partes. Presentamos un modelo de energía basado en los perfiles energéticos de diferentes tipos de aplicaciones y sus combinaciones. Basado en este modelo, diseñamos estrategias sofisticadas de consolidación que tienen en cuenta las características generales de cada aplicación y las distribuyen equitativamente en los recursos para evitar conflictos.

Presentamos perfiles de energía precisos para cada tipo de aplicación y su combinación para reducir el error entre la estimación y el consumo de energía en un procesador Xeon IvyBridge.

Para obtener los perfiles de energía, llevamos a cabo un conjunto de evaluaciones empíricas para cada tipo de trabajo, utilizando el *benchmark* SysBench. Estas evaluaciones se realizan en un servidor express x3650 M4 con un procesador Xeon IvyBridge E5-2650v2 y un sistema operativo CentOS 7 Linux.

Usamos dos escenarios de contención con respecto al uso de la memoria total del servidor. En el primero, las aplicaciones usan el 100% y en el segundo escenario, requieren 107%. En el primer escenario, el tiempo total de ejecución se incrementa casi tres veces en comparación con el tiempo del usuario (60 segundos). En el segundo escenario, el tiempo total de ejecución se incrementa casi 20 veces. Los efectos de contención ocurren porque el sistema requiere más tiempo en paginación en lugar de hacer las ejecuciones. En ambos escenarios, el consumo de energía es mayor en comparación cuando se evita la contención.

Para demostrar la solidez de nuestro modelo energético, llevamos a cabo mediciones de consumo de energía en los niveles de CPU y servidor. Los resultados de las mediciones muestran que cada tipo de tarea proporciona un perfil de energía diferente. El nivel de energía consumido por el procesador se incrementa en un factor constante con respecto al nivel consumido por el servidor. Para las aplicaciones intensivas en memoria y CPU, las mediciones a nivel del servidor son aproximadamente 2.4x los valores obtenidos a nivel de procesador. En modo inactivo, el factor de incremento es de 10x.

Hasta donde llega nuestro conocimiento, este trabajo es el primero en la literatura que considera la asignación de recursos teniendo en cuenta el tipo de aplicación y su concentración cuando se utilizan cargas de trabajo heterogéneas. El consumo de energía se considera como una función del uso del CPU y la concentración del tipo de trabajos.

En base al modelo de energía propuesto y enfocándonos en una calendarización de dos niveles. Diseñamos estrategias conscientes de la energía, de la utilización del CPU y del tipo de trabajo. Comparamos las estrategias propuestas con estrategias de asignación clásicas.

Todos los experimentos se realizaron utilizando CloudSim, un sistema para el modelado y la simulación de infraestructuras y servicios de computación en la nube.

Para proporcionar una guía adecuada al momento de elegir la mejor estrategia, realizamos un análisis del consumo de energía de acuerdo con la metodología de degradación. Para eliminar la influencia de una pequeña porción de datos con una desviación significativa en el proceso de evaluación comparativa, y ayudar con la interpretación de los datos, presentamos los perfiles de desempeño de nuestras estrategias. La estrategia que maximiza la utilización y realiza un balance entre los tipos de tareas "*MaxU_MinC"* presenta la mejor eficiencia energética.

Existen una serie de desafíos de investigación que deben ser abordados. Queremos evaluar las estrategias de asignación con más tipos de trabajo, por ejemplo, uso intensivo de red (NI), uso intensivo de disco y analizar el consumo de energía al combinarlos en diferentes escenarios. Es importante estudiar los perfiles energéticos de cada trabajo y las estrategias de consolidación en diferentes arquitecturas.

Finalmente, queremos evaluar la eficiencia energética de nuestras políticas de calendarización en diferentes entornos de nube, como los centros de datos de nube en contenedores.

Literatura citada

Armenta-Cano, F. A., Tchernykh, A., Cortes-Mendoza, J. M., Yahyapour, R., Drozdov, A. Y., Bouvry, P., Nesmachnow, S. 2017. Min_c: Heterogeneous concentration policy for energy-aware scheduling of jobs with resource contention. *Programming and Computer Software*, *43*(3). https://doi.org/10.1134/S0361768817030021

Bailey, D. H., Barszcz, E., Dagum, L., Simon, H. D. 1993. NAS parallel benchmark results. *IEEE Parallel & Distributed Technology: Systems & Applications*, *1*(1), 43–51. https://doi.org/10.1109/88.219861

Bellosa, F. 2000. The benefits of event. In *Proceedings of the 9th workshop on ACM SIGOPS European workshop beyond the PC: new challenges for the operating system - EW 9* (p. 37). New York, New York, USA: ACM Press. https://doi.org/10.1145/566726.566736

Beloglazov, A., Abawajy, J., Buyya, R. 2012. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Future Generation Computer Systems*, *28*(5), 755–768. https://doi.org/10.1016/j.future.2011.04.017

Blagodurov, S., Fedorova, A. 2012. Towards the contention aware scheduling in HPC cluster environment. *Journal of Physics: Conference Series*, *385*(1), 12010. https://doi.org/10.1088/1742-6596/385/1/012010

Bui, V. Q. B., Teabe, B., Tchana, A., Hagimont, D. 2011. Kyoto. In *Proceedings of the International workshop on Virtualization Technologies - VT15* (pp. 1–6). New York, New York, USA: ACM Press. https://doi.org/10.1145/2835075.2835077

Buyya, R., Murshed, M. 2002. GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing. *Concurrency and Computation: Practice and Experience*, *14*(13–15), 1175–1220. https://doi.org/10.1002/cpe.710

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., Buyya, R. 2011. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Softw. Pract. Exper.*, *41*(1), 23–50. https://doi.org/10.1002/spe.995

Calheiros, R. N., Ranjan, R., De Rose, C. A. F., Buyya, R. 2009. CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services. Retrieved from http://arxiv.org/abs/0903.2525

Chuvakin, A. A., Schmidt, K. J., Phillips, C., Moulder, P. 2013. *Logging and log management : the authoritative guide to understanding the concepts surrounding logging and log management*. Syngress.

Combarro, M., Tchernykh, A., Kliazovich, D., Drozdov, A., Radchenko, G. 2016. Energy-Aware Scheduling with Computing and Data Consolidation Balance in 3-Tier Data Center. In *2016 International Conference on Engineering and Telecommunication (EnT)* (pp. 29–33). IEEE. https://doi.org/10.1109/EnT.2016.015

Delforge, P., Whitney, J. 2014. *Data Center Efficiency Assessment*. Retrieved from www.suerossi.com

Dixit, K. M. 1991. The SPEC benchmarks. *Parallel Computing*, *17*(10–11), 1195–1209. https://doi.org/10.1016/S0167-8191(05)80033-X

Dongarra, J. J., Luszczek, P., Petitet, A. 2003. The LINPACK Benchmark: past, present and future. *Concurrency and Computation: Practice and Experience*, *15*(9), 803–820. https://doi.org/10.1002/cpe.728

Dorronsoro, B., Nesmachnow, S., Taheri, J., Zomaya, A. Y., Talbi, E.-G., Bouvry, P. 2014. A hierarchical

approach for energy-efficient scheduling of large workloads in multicore distributed systems. *Sustainable Computing: Informatics and Systems*, *4*(4), 252–261. https://doi.org/10.1016/j.suscom.2014.08.003

Dugan, J., Estabrook, J., Ferbuson, J., Gallatin, A., Gates, M., Gibbs, K., … Warshavsky, A. (n.d.). iPerf - The TCP, UDP and SCTP network bandwidth measurement tool. Retrieved July 27, 2017, from https://iperf.fr/

Dumitrescu, C. L., Foster, I. 2005. GangSim: a simulator for grid scheduling studies. In *IEEE International Symposium on Cluster Computing and the Grid, 2005. CCGrid 2005* (Vol. 2, p. 1151–1158 Vol. 2). https://doi.org/10.1109/CCGRID.2005.1558689

Eckart, Z. 1999. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. Swiss Federal Institute of Technology Zurich. Retrieved from http://www.tik.ee.ethz.ch/~sop/publicationListFiles/zitz1999a.pdf

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. https://doi.org/10.1016/j.jpdc.2014.06.013

Fortino, G. 2013. *Intelligent distributed computing VI : Proceedings of the 6th International Symposium on Intelligent Distributed Computing-- IDC 2012, Calabria, Italy, September 2012*. Springer.

Gao, Y., Yanzhi Wang, Gupta, S. K., Pedram, M. 2013. An energy and deadline aware resource provisioning, scheduling and optimization framework for cloud systems. In *2013 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)* (pp. 1–10). IEEE. https://doi.org/10.1109/CODES-ISSS.2013.6659018

Graham, R. L., Lawler, E. L., Lenstra, J. K., Kan, A. H. G. R. 1979. Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey (pp. 287–326). https://doi.org/10.1016/S0167-5060(08)70356-X

Hosseinimotlagh, S., Khunjush, F., Hosseinimotlagh, S. 2014. A Cooperative Two-Tier Energy-Aware Scheduling for Real-Time Tasks in Computing Clouds. In *2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing* (pp. 178–182). IEEE. https://doi.org/10.1109/PDP.2014.91

Hsu, C.-H., Slagter, K. D., Chen, S.-C., Chung, Y.-C. 2014. Optimizing Energy Consumption with Task Consolidation in Clouds. *Information Sciences*, *258*, 452–462. https://doi.org/10.1016/j.ins.2012.10.041

Jin, H., Qin, H., Wu, S., Guo, X. 2015. CCAP: A Cache Contention-Aware Virtual Machine Placement Approach for HPC Cloud. *International Journal of Parallel Programming*, *43*(3), 403–420. https://doi.org/10.1007/s10766-013-0286-1

Jokanovic, A., Rodriguez, G., Sancho, J. C., Labarta, J. 2010. Impact of Inter-application Contention in Current and Future HPC Systems. In *2010 18th IEEE Symposium on High Performance Interconnects* (pp. 15–24). IEEE. https://doi.org/10.1109/HOTI.2010.25

Kaplan, J. M., Forrest, W., Kindler, N. 2008. *Revolutionizing Data Center Energy Efficiency*. SanFrancisco,CA. Retrieved from https://www.sallan.org/pdf-docs/McKinsey_Data_Center_Efficiency.pdf

Kliazovich, D., Pecero, J. E., Tchernykh, A., Bouvry, P., Khan, S. U., Zomaya, A. Y. 2016. CA-DAG: Modeling Communication-Aware Applications for Scheduling in Cloud Computing. *Journal of Grid Computing*, *14*(1), 23–39. https://doi.org/10.1007/s10723-015-9337-8

Kopytov, A. 2017. Sysbench. Retrieved July 11, 2017, from https://github.com/akopytov/sysbench

Legrand, A., Marchal, L., Casanova, H. 2003. Scheduling distributed applications: the SimGrid simulation framework. In *3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2003. Proceedings. CCGrid 2003* (pp. 138–145). https://doi.org/10.1109/CCGRID.2003.1199362

Li, Y., Lu, X., Miller, E. L., Long, D. D. E. 2015. ASCAR: Automating contention management for high-performance storage systems. In *2015 31st Symposium on Mass Storage Systems and Technologies (MSST)* (pp. 1–16). IEEE. https://doi.org/10.1109/MSST.2015.7208287

Liang Luo, Wenjun Wu, Dichen Di, Fei Zhang, Yizhou Yan, Yaokuan Mao. 2012. A resource scheduling algorithm of cloud computing based on energy efficient optimization methods. In *2012 International Green Computing Conference (IGCC)* (pp. 1–6). IEEE. https://doi.org/10.1109/IGCC.2012.6322251

Liu, Z., Ma, R., Zhou, F., Yang, Y., Qi, Z., Guan, H. 2010. Power-aware I/O-Intensive and CPU-Intensive applications hybrid deployment within virtualization environments. In *2010 IEEE International Conference on Progress in Informatics and Computing (PIC)* (Vol. 1, pp. 509–513). https://doi.org/10.1109/PIC.2010.5687570

Loewe, W., McLarty, T., Morrone, C. 2012. Ior benchmark.

Luo, J., Li, X., Chen, M. 2014. Hybrid shuffled frog leaping algorithm for energy-efficient dynamic consolidation of virtual machines in cloud data centers. *Expert Systems with Applications*, *41*(13), 5804–5816. https://doi.org/10.1016/j.eswa.2014.03.039

Luo, L., Wu, W., Tsai, W. T., Di, D., Zhang, F. 2013. Simulation of power consumption of cloud data centers. *Simulation Modelling Practice and Theory*, *39*, 152–171. https://doi.org/10.1016/j.simpat.2013.08.004

McCalpin, J. D. 2005. The STREAM Benchmark. Retrieved from http://www.cs.virginia.edu/~mccalpin/STREAM_Benchmark_2005-01-25.pdf

Mell, P., Grance, T. 2011. The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology. Retrieved from http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf

Michael Feldman. 2017. TOP500 Meanderings: Supercomputers Take Big Green Leap in 2017 | TOP500 Supercomputer Sites. Retrieved October 22, 2017, from https://www.top500.org/news/top500-meanderings-supercomputers-take-big-green-leap-in-2017/

Norcott, W., Capps, D. 2003. Iozone filesystem benchmark.

Piraghaj, S. F., Calheiros, R. N., Chan, J., Dastjerdi, A. V., Buyya, R. 2016. Virtual machine customization and task mapping architecture for efficient allocation of cloud data center resources. *Computer Journal*, *59*(2), 208–224. https://doi.org/10.1093/comjnl/bxv106

Ramírez-Alcaraz, J. M., Tchernykh, A., Yahyapour, R., Schwiegelshohn, U., Quezada-Pina, A., González-García, J. L., Hirales-Carbajal, A. 2011. Job Allocation Strategies with User Run Time Estimates for Online Scheduling in Hierarchical Grids. *Journal of Grid Computing*, *9*(1), 95–116. https://doi.org/10.1007/s10723-011-9179-y

Rodríguez Díaz, A., Tchernykh, A., Ecker, K. H. 2003. Algorithms for dynamic scheduling of unit execution time tasks. *European Journal of Operational Research*, *146*(2), 403–416. https://doi.org/10.1016/S0377-2217(02)00236-9

Roytman, A., Kansal, A., Govindan, S., Liu, J., Nath, S. 2013. PACMan: Performance Aware Virtual Machine Consolidation. *ICAC*. Retrieved from https://www.usenix.org/system/files/tech-schedule/icac13_full_proceedings.pdf#page=94

Switched PDU with Per Outlet Power Metering Outlet Metered Reboot Switch. 2017. Retrieved July 31,

2017, from http://www.wti.com/specpdf/specs_outlet-metered-pdu.pdf

Tang, G., Jiang, W., Xu, Z., Liu, F., Wu, K. 2015. Zero-Cost, Fine-Grained Power Monitoring of Datacenters Using Non-Intrusive Power Disaggregation. In *Proceedings of the 16th Annual Middleware Conference on - Middleware '15* (pp. 271–282). New York, New York, USA: ACM Press. https://doi.org/10.1145/2814576.2814728

Tchernykh, A., Lozano, L., Bouvry, P., Pecero, J. E., Schwiegelshohn, U., Nesmachnow, S. 2014. Energy-aware online scheduling: Ensuring quality of service for IaaS clouds. In *2014 International Conference on High Performance Computing & Simulation (HPCS)* (pp. 911–918). IEEE. https://doi.org/10.1109/HPCSim.2014.6903786

Tchernykh, A., Lozano, L., Schwiegelshohn, U., Bouvry, P., Pecero, J. E., Nesmachnow, S., Drozdov, A. Y. 2016. Online Bi-Objective Scheduling for IaaS Clouds Ensuring Quality of Service. *Journal of Grid Computing*, *14*(1), 5–22. https://doi.org/10.1007/s10723-015-9340-0

Tchernykh, A., Pecero, J. E., Barrondo, A., Schaeffer, E. 2014. Adaptive energy efficient scheduling in Peer-to-Peer desktop grids. *Future Generation Computer Systems*, *36*, 209–220. https://doi.org/10.1016/j.future.2013.07.011

Tchernykh, A., Ramírez, J. M., Avetisyan, A., Kuzjurin, N., Grushin, D., Zhuk, S. 2006. Two Level Job-Scheduling Strategies for a Computational Grid (pp. 774–781). Springer, Berlin, Heidelberg. https://doi.org/10.1007/11752578_93

Tchernykh, A., Schwiegelshohn, U., Yahyapour, R., Kuzjurin, N. 2010. On-line hierarchical job scheduling on grids with admissible allocation. *Journal of Scheduling*, *13*(5), 545–552. https://doi.org/10.1007/s10951-010-0169-x

The Grid Workloads Archive. (n.d.). Retrieved July 11, 2017, from http://gwa.ewi.tudelft.nl/

Treibig, J., Hager, G., Wellein, G. 2010. LIKWID: A Lightweight Performance-Oriented Tool Suite for x86 Multicore Environments. In *2010 39th International Conference on Parallel Processing Workshops* (pp. 207–216). IEEE. https://doi.org/10.1109/ICPPW.2010.38

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. https://doi.org/10.1109/TPDS.2007.70606

Tudor, B. M., Teo, Y. M. 2013. On understanding the energy consumption of ARM-based multicore servers. In *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems - SIGMETRICS '13* (Vol. 41, p. 267). New York, New York, USA: ACM Press. https://doi.org/10.1145/2465529.2465553

Ubench. (n.d.). Retrieved July 27, 2017, from http://phystech.com/download/ubench.html

Wang, X., Liu, X., Fan, L., Jia, X. 2013. A Decentralized Virtual Machine Migration Approach of Data Centers for Cloud Computing. *Mathematical Problems in Engineering*, *2013*, 1–10. https://doi.org/10.1155/2013/878542

Apéndice

A.1 Resultados del experimento A.

Tabla 10. Consumo total de energía de las 10 estrategias durante 30 semanas

Semana	FFit	Max_u	MaxU_MinC	Min_e	Min_c	Min_ujt	Min_u	MinU_MinC	Rand	RR
W1	260606860	260269936	266171451	330485386	300010621	324167328	338162653	320781915	289371761	297104801
W2	253847419	257061212	254785981	300631532	286086707	294839291	309954459	297303309	288048583	277991848
W3	347037329	346243442	345436994	389279660	387290490	398397004	395476749	394985371	374892860	368259797
W4	433587749	431659498	431320605	487545678	473356625	500733106	493582107	498431469	463894460	462684531
W5	387896776	383901562	389053531	449531068	439906307	462334465	463809688	462641456	412028513	406738054
W6	326102018	327018909	328270960	372950191	361036162	390093322	390718710	384681562	351447052	354527308
W7	309287154	313247659	311947953	375744437	373307330	384726537	381827396	388472654	334318703	363902089
W8	392469812	394827708	394651184	437235636	429784839	439876438	450464148	443923078	420283145	410190605
W9	407865104	408767685	416336492	440926986	432862336	448154702	438756418	447777997	431796112	426405258
W10	321186310	315187473	322772062	362137552	368208243	371634100	373271786	370673495	344143168	353731203
W11	407045848	406394186	411819238	479326634	448234617	470663560	488098928	484097420	438279928	436547721
W12	378659732	378547516	373974283	435686158	424704669	436636361	444370614	448524772	408488552	418491543
W13	417235532	413973407	416129129	502668252	475113924	501990597	516421932	513160236	445235983	464622239
W14	368765179	368870867	371740362	397598980	389760681	395100449	398846868	392191472	380359905	380682297
W15	420868079	424588628	423017702	457695291	447615337	452450736	460475267	455282520	442569759	442791096
W16	296857052	294529454	295645964	344263419	328808779	351058570	349386578	339997383	317861583	295513154
W17	339162541	337411149	345210267	392584142	353275414	399689009	414970949	382676383	370619025	363859132
W18	296943023	295442728	299199507	323803278	319729037	329741675	324962980	322079106	311593070	307322957
W19	374108715	379359094	376234493	437033849	419639887	428670183	432519573	427581892	397130359	403144897
W20	420472816	420592340	419485122	503899491	462911686	488311055	501274376	489971372	467094462	471859812
W21	404681213	406656716	407149150	545070777	509430342	522014299	533724424	530108853	462032460	465806250
W22	377881567	385009115	388346682	481242713	496753512	517277371	521268829	511766411	458991657	458148775
W23	280511960	283584842	283969986	319258343	299787336	316680524	319013698	314353925	298140786	296349729
W24	343422773	343974883	343360095	371286466	370552163	383944937	375222662	381906847	361493350	372602253
W25	354926527	347859625	351016979	390364155	385009505	387178807	396878662	391876671	378072021	383768194
W26	419331459	416176254	416786921	454971456	442489128	466945637	476941906	465784859	449696296	442344457
W27	368855387	367820692	372824336	424286690	403668562	427291001	441164352	428233973	388003751	407425088
W28	398203587	384309367	385934546	446800116	426172134	447125698	444688968	448111321	420351633	417444726
W29	499759331	502596140	504481612	594381362	563970616	583126749	590971620	599221269	533897077	534787035
W30	447707735	455343603	457410438	543680574	512703347	528514604	534732010	539447922	495175082	489602944

Tabla 11. Consumo total de energía en kWh de las 10 estrategias durante 30 semanas

Semanas	FFit	Max_u	MaxU_MinC	Min_e	Min_c	Min_ujt	Min_u	MinU_MinC	Rand	RR
W1	72.3907944	72.2972046	73.9365141	91.801496	83.3362835	90.04648	93.9340702	89.1060875	80.3810446	82.5291114
W2	70.5131721	71.4058923	70.7738837	83.508759	79.4685297	81.899803	86.0984608	82.5842524	80.0134954	77.2199577
W3	96.3992581	96.1787338	95.9547204	108.133239	107.580692	110.665834	109.854652	109.718159	104.136906	102.294388
W4	120.441041	119.905416	119.811279	135.429355	131.487951	139.09253	137.106141	138.453186	128.859572	128.523481
W5	107.749104	106.639323	108.070425	124.869741	122.196196	128.42624	128.836024	128.511516	114.452365	112.982793
W6	90.5838939	90.8385858	91.1863779	103.597275	100.287823	108.359256	108.532975	106.855989	97.624181	98.4798079
W7	85.9130983	87.0132385	86.6522091	104.373455	103.69648	106.868482	106.063166	107.909071	92.8663064	101.083914
W8	109.019392	109.674363	109.625329	121.454343	119.384678	122.187899	125.12893	123.311966	116.745318	113.941835
W9	113.295862	113.546579	115.649025	122.479718	120.239538	124.487417	121.876783	124.382777	119.943364	118.445905
W10	89.2184194	87.5520759	89.658906	100.593765	102.280067	103.231694	103.686607	102.96486	95.5953244	98.2586674
W11	113.068291	112.887274	114.394233	133.146287	124.509616	130.739878	135.583036	134.471506	121.744425	121.263256
W12	105.183259	105.152088	103.881745	121.023933	117.973519	121.287878	123.436282	124.590214	113.469042	116.247651
W13	115.898759	114.992613	115.591425	139.63007	131.97609	139.441832	143.450537	142.54451	123.676662	129.061733
W14	102.434772	102.46413	103.261212	110.444161	108.266856	109.750125	110.790797	108.942075	105.655529	105.745083
W15	116.9078	117.941286	117.504917	127.137581	124.337593	125.68076	127.909796	126.467367	122.936044	122.997527
W16	82.4602921	81.8137371	82.123879	95.6287275	91.3357721	97.5162693	97.0518271	94.4437176	88.2948843	82.0869872
W17	94.2118169	93.7253192	95.8917409	109.051151	98.1320595	111.024725	115.269708	106.298995	102.949729	101.071981
W18	82.484173	82.0674244	83.1109743	89.9453549	88.8136213	91.5949098	90.2674945	89.4664183	86.5536306	85.367488
W19	103.919088	105.377526	104.509581	121.398291	116.566635	119.075051	120.144326	118.772748	110.313989	111.984694
W20	116.798004	116.831206	116.523645	139.972081	128.586579	135.64196	139.242882	136.103159	129.748462	131.07217
W21	112.411448	112.960199	113.096986	151.408549	141.508428	145.003972	148.256784	147.252459	128.34235	129.390625
W22	104.967102	106.946977	107.874078	133.678531	137.987087	143.688159	144.796897	142.157336	127.497683	127.263549
W23	77.9199888	78.7735671	78.8805516	88.6828731	83.2742599	87.9668123	88.6149161	87.3205348	82.8168849	82.3193691
W24	95.3952148	95.5485786	95.3778042	103.135129	102.931156	106.651371	104.228517	106.085235	100.414819	103.500626
W25	98.590702	96.6276736	97.5047163	108.434488	106.947085	107.549669	110.244073	108.854631	105.020006	106.602276
W26	116.480961	115.604515	115.774145	126.38096	122.913647	129.707121	132.483863	129.384683	124.915638	122.87346
W27	102.45983	102.172415	103.562316	117.857414	112.130156	118.691945	122.545653	118.953881	107.77882	113.173636
W28	110.612108	106.752602	107.204041	124.111143	118.381148	124.201583	123.524713	124.475367	116.764343	115.956868
W29	138.822036	139.610039	140.133781	165.105934	156.658504	161.979652	164.158783	166.450353	148.304744	148.551954
W30	124.36326	126.484334	127.058455	151.022382	142.417596	146.809612	148.536669	149.846645	137.548634	136.000818

Tabla 12. Perfil de desempeño de las 10 estrategias

δ	FFit	Max_u	MaxU_MinC	Min_e	Min_c	Min_ujt	Min_u	MinU_MinC	Rand	RR
0	0	0	0	0	0	0	0	0	0	0
0.02	0.933	1	0.8	0	0	0	0	0	0	0.033
0.04	1	1	1	0	0	0	0	0	0.033	0.066
0.06	1	1	1	0	0.066	0	0	0	0.2	0.266
0.08	1	1	1	0.033	0.233	0.066	0.033	0.033	0.533	0.5
0.1	1	1	1	0.2	0.366	0.1	0.166	0.133	0.8	0.666
0.12	1	1	1	0.233	0.566	0.2	0.166	0.2	0.9	0.766
0.14	1	1	1	0.366	0.733	0.3	0.2	0.333	0.933	0.866
0.16	1	1	1	0.466	0.866	0.4	0.433	0.466	0.966	0.933
0.18	1	1	1	0.7	0.9	0.666	0.433	0.666	0.966	0.966
0.2	1	1	1	0.766	0.9	0.8	0.7	0.766	0.966	0.966
0.22	1	1	1	0.9	0.933	0.866	0.766	0.833	1	1
0.24	1	1	1	0.9	0.933	0.866	0.866	0.9	1	1
0.26	1	1	1	0.9	0.966	0.933	0.9	0.933	1	1
0.28	1	1	1	0.966	0.966	0.933	0.9	0.933	1	1
0.3	1	1	1	0.966	0.966	0.966	0.933	0.933	1	1
0.32	1	1	1	0.966	1	0.966	0.966	0.966	1	1
0.34	1	1	1	0.9666	1	0.966	0.966	0.966	1	1
0.36	1	1	1	1	1	0.966	0.966	1	1	1
0.38	1	1	1	1	1	1	1	1	1	1

A.2 Resultados del experimento B.

Tabla 13. Consumo total de energía de las 10 estrategias durante 30 semanas

Semanas	FFit	Max_u	MaxU_MinC	Min_e	Min_c	Min_ujt	Min_u	MinU_MinC	Rand	RR
W1	260606860	260269936	266171451	330485386	300010621	324167328	338162653	320781915	289371761	297104801
W2	253847419	257061212	254785981	300631532	286086707	294839291	309954459	297303309	288048583	277991848
W3	347037329	346243442	345436994	389279660	387290490	398397004	395476749	394985371	374892860	368259797
W4	433587749	431659498	431320605	487545678	473356625	500733106	493582107	498431469	463894460	462684531
W5	387896776	383901562	389053531	449531068	439906307	462334465	463809688	462641456	412028513	406738054
W6	326102018	327018909	328270960	372950191	361036162	390093322	390718710	384681562	351447052	354527308
W7	309287154	313247659	311947953	375744437	373307330	384726537	381827396	388472654	334318703	363902089
W8	392469812	394827708	394651184	437235636	429784839	439876438	450464148	443923078	420283145	410190605
W9	407865104	408767685	416336492	440926986	432862336	448154702	438756418	447777997	431796112	426405258
W10	321186310	315187473	322772062	362137552	368208243	371634100	373271786	370673495	344143168	353731203
W11	407045848	406394186	411819238	479326634	448234617	470663560	488098928	484097420	438279928	436547721
W12	378659732	378547516	373974283	435686158	424704669	436636361	444370614	448524772	408488552	418491543
W13	417235532	413973407	416129129	502668252	475113924	501990597	516421932	513160236	445235983	464622239
W14	368765179	368870867	371740362	397598980	389760681	395100449	398846868	392191472	380359905	380682297
W15	420868079	424588628	423017702	457695291	447615337	452450736	460475267	455282520	442569759	442791096
W16	296857052	294529454	295645964	344263419	328808779	351058570	349386578	339997383	317861583	295513154
W17	339162541	337411149	345210267	392584142	353275414	399689009	414970949	382676383	370619025	363859132
W18	296943023	295442728	299199507	323803278	319729037	329741675	324962980	322079106	311593070	307322957
W19	374108715	379359094	376234493	437033849	419639887	428670183	432519573	427581892	397130359	403144897
W20	420472816	420592340	419485122	503899491	462911686	488311055	501274376	489971372	467094462	471859812
W21	404681213	406656716	407149150	545070777	509430342	522014299	533724424	530108853	462032460	465806250
W22	377881567	385009115	388346682	481242713	496753512	517277371	521268829	511766411	458991657	458148775
W23	280511960	283584842	283969986	319258343	299787336	316680524	319013698	314353925	298140786	296349729
W24	343422773	343974883	343360095	371286466	370552163	383944937	375222662	381906847	361493350	372602253
W25	354926527	347859625	351016979	390364155	385009505	387178807	396878662	391876671	378072021	383768194
W26	419331459	416176254	416786921	454971456	442489128	466945637	476941906	465784859	449696296	442344457
W27	368855387	367820692	372824336	424286690	403668562	427291001	441164352	428233973	388003751	407425088
W28	398203587	384309367	385934546	446800116	426172134	447125698	444688968	448111321	420351633	417444726
W29	499759331	502596140	504481612	594381362	563970616	583126749	590971620	599221269	533897077	534787035
W30	447707735	455343603	457410438	543680574	512703347	528514604	534732010	539447922	495175082	489602944

Tabla 14. Consumo total de energía en kWh de las 10 estrategias durante 30 semanas

Semanas	FFit	Max_u	MaxU_MinC	Min_e	Min_c	Min_ujt	Min_u	MinU_MinC	Rand	RR
W1	81.7639235	83.3011946	74.4040729	95.1225669	89.0835323	92.306186	96.5383288	91.5888484	85.5268516	87.4266053
W2	75.6783382	78.5263166	71.2157387	85.3312123	80.9859379	83.2620648	87.536899	83.9347653	83.8568661	79.9195942
W3	101.688896	101.591818	96.6174433	111.085822	108.178417	111.187456	112.636309	110.264377	106.406795	105.301458
W4	124.781074	123.748548	120.313934	133.978217	132.863906	140.069544	137.591949	139.8274	131.72464	131.434084
W5	114.796628	113.299664	108.588172	123.976534	123.286664	129.465247	129.945593	129.571916	116.052672	114.375156
W6	91.4495816	91.6611366	91.5104571	107.559173	100.700829	108.771869	108.873703	107.170552	98.3399856	98.8271798
W7	89.083989	90.508331	87.2122687	106.052648	104.174642	107.315236	106.725241	108.723323	98.2644306	103.482804
W8	112.920522	114.198524	110.185471	124.794576	120.062238	122.716842	125.97135	123.920927	119.018217	115.387652
W9	123.523752	125.982126	116.424569	125.138584	121.533087	125.41667	123.573837	125.754787	127.753104	124.138909
W10	93.5921559	94.0209004	90.1502969	100.011228	103.735596	103.966881	104.459395	103.713829	98.0725418	100.851061
W11	123.077154	122.920519	115.063374	133.311965	126.601129	132.727431	137.868122	136.472238	125.776899	124.73419
W12	111.286978	114.064435	104.561596	127.820819	118.520332	122.021674	124.996182	126.052312	115.315679	117.395527
W13	136.168237	138.153426	116.407611	149.83779	142.93608	146.589786	151.583978	150.566466	138.483092	141.519825
W14	121.208311	122.783416	104.043141	115.851199	111.713454	113.089694	116.378291	112.286348	121.069331	119.263385
W15	147.435529	146.412959	118.557909	136.941287	129.739278	131.010126	134.708136	132.141619	135.338905	134.566924
W16	85.4196031	85.359599	82.639078	95.2477107	91.7514576	97.8843433	97.5695625	94.8939167	89.3114414	83.5133417
W17	97.4530246	97.0714464	96.4025349	111.72578	98.9300447	111.566866	116.069656	106.983541	104.621645	103.088388
W18	85.8663035	87.8300599	83.6372341	89.4744316	89.2906571	91.9799472	90.8427677	89.8707802	87.9202645	86.4728257
W19	112.335711	114.302492	105.201629	119.555564	118.979066	121.408832	122.851787	121.440891	114.989427	115.560194
W20	121.193184	122.078679	117.066671	136.036802	129.509467	136.356761	141.034579	136.888515	132.201554	132.944374
W21	125.531247	122.466918	113.812091	153.420205	143.471385	146.677327	150.19444	149.061472	132.613352	134.116632
W22	116.461356	114.522463	108.543297	140.713186	139.01347	144.716888	145.915817	143.32868	132.166677	130.492542
W23	82.0122928	82.5642409	79.4097114	87.7123986	84.1553153	88.4092947	89.1552506	87.762273	85.6094394	83.7172473
W24	100.744634	99.8540721	96.0427287	106.147391	103.525918	107.270492	104.949834	106.596536	104.980348	105.096116
W25	111.599422	109.24758	98.2928275	111.032436	107.710045	108.219501	111.833989	109.538225	108.932293	111.8107
W26	134.215814	132.35677	116.608525	131.317021	124.261809	130.333711	133.514872	130.107138	127.631492	126.746218
W27	112.588952	111.320094	104.244985	120.01038	112.723786	119.229341	124.009577	119.592516	110.842549	115.20453
W28	119.091499	112.497334	107.821079	122.551478	118.967562	124.683153	124.045703	124.956227	120.115564	119.190457
W29	148.630217	149.494896	141.120284	168.640516	157.566839	162.69996	165.177419	167.173131	150.930688	151.252648
W30	147.673659	155.848378	128.045211	156.621337	148.774502	151.080281	154.316742	152.473469	155.334714	146.078043

Tabla 15. Perfil de desempeño de las 10 estrategias

δ	FFit	Max_u	MaxU_MinC	Min_e	Min_c	Min_ujt	Min_u	MinU_MinC	Rand	RR
0	0	0	0	0	0	0	0	0	0	0
0.02	0.066	0.066	1	0	0	0	0	0	0	0.033
0.04	0.333	0.266	1	0	0.033	0	0	0	0	0.066
0.06	0.466	0.533	1	0	0.1	0	0	0	0.033	0.166
0.08	0.666	0.633	1	0.066	0.233	0.033	0.033	0.066	0.2	0.266
0.1	0.733	0.733	1	0.066	0.366	0.1	0.1	0.1	0.533	0.5
0.12	0.8	0.833	1	0.233	0.633	0.3	0.133	0.3	0.666	0.6
0.14	0.833	0.866	1	0.4	0.766	0.3	0.233	0.333	0.733	0.766
0.16	0.9	0.866	1	0.633	0.8	0.566	0.4	0.533	0.8	0.833
0.18	0.966	0.866	1	0.7	0.833	0.733	0.5	0.666	0.9	0.9
0.2	0.966	0.933	1	0.766	0.9	0.833	0.7	0.8	0.933	0.933
0.22	0.966	0.966	1	0.8	0.9	0.833	0.8	0.833	1	1
0.24	0.966	1	1	0.866	0.933	0.866	0.866	0.866	1	1
0.26	1	1	1	0.866	0.933	0.933	0.866	0.9	1	1
0.28	1	1	1	0.9	0.966	0.933	0.866	0.9	1	1
0.3	1	1	1	0.966	1	0.966	0.9	0.933	1	1
0.32	1	1	1	0.966	1	0.966	0.966	0.966	1	1
0.34	1	1	1	0.966	1	1	0.966	1	1	1
0.36	1	1	1	1	1	1	1	1	1	1

A.3 Resultados del experimento C.

Tabla 16. Consumo total de energía de las 10 estrategias durante 30 semanas

Semanas	FFit	Max_u	MaxU_MinC	Min_e	Min_c	Min_ujt	Min_u	MinU_MinC	Rand	RR
W1	750309930	752656330	752735923	989632457	884006634	966321134	1008333904	952729909	848514495	874275199
W2	722351200	735349031	716104501	880563822	835849797	873723251	921625501	881523309	846110446	809155445
W3	987397389	981236600	970953423	1143275241	1126297226	1171049820	1166248119	1160445599	1086590567	1069131991
W4	1208531782	1200287335	1194632653	1395726325	1364399154	1462527650	1434011375	1457494803	1325566009	1322296414
W5	1088687811	1070405357	1076210001	1332766027	1270169178	1359592240	1358925154	1358147021	1173290231	1174563500
W6	912307683	911628474	915471131	1105621717	1050074074	1154944842	1150587304	1136918308	1012801011	1020622572
W7	868882521	880234222	870161300	1082201230	1093844855	1147522097	1134418821	1159158695	974693799	1069105785
W8	1094368393	1101623552	1093859847	1249717583	1235187724	1290461124	1327484582	1303842283	1209254297	1171084380
W9	1160592950	1158203038	1163241445	1290824083	1251267799	1317609800	1289904286	1319351591	1252860498	1235734737
W10	904340220	886748520	898591186	1046960027	1065264731	1086986192	1092238524	1083789592	987665724	1022643707
W11	1149655238	1145344716	1144132969	1334138528	1291124705	1381147345	1437845969	1426371111	1260304434	1256573424
W12	1063334421	1064124102	1033505162	1282982978	1230334854	1283438059	1311282771	1325337647	1171074021	1207268810
W13	1208196175	1198346494	1166879988	1504381642	1404731800	1495854113	1540297467	1534108186	1307911951	1371681282
W14	1066030149	1063683967	1038028130	1172074304	1120762245	1147489991	1170121040	1140393767	1111357165	1108332792
W15	1232586170	1235466566	1182511221	1339816891	1294697008	1317515916	1352337065	1331436580	1287681826	1285411677
W16	834138118	823962187	822053287	998156265	951459728	1036809924	1030335867	1000392667	913243847	858762815
W17	953977910	946404738	965873909	1165712135	1018034495	1182142832	1225523343	1130402151	1075131558	1051950553
W18	834119253	830959174	832616901	925122564	921066499	965670771	951545554	941048624	892711876	879637554
W19	1055646030	1069788059	1044279293	1227625940	1216192299	1257934251	1275432750	1258646559	1141595569	1164138033
W20	1177959042	1174243748	1165772618	1434352817	1337673423	1438451315	1484014812	1447523684	1352644746	1369028554
W21	1147178462	1145708269	1131996169	1585948094	1503859383	1561658287	1591793309	1588085259	1344819319	1357402191
W22	1077946479	1089617211	1089852750	1499274138	1464574957	1539809431	1545838404	1520043232	1348113490	1344853282
W23	790283951	796431004	792967343	916770758	860298692	925354939	935782823	919461231	859040630	849688497
W24	962116908	961009005	950824200	1097987159	1060699605	1117423158	1092259691	1113030261	1037112527	1069886414
W25	1018378857	991749364	981145734	1147373031	1113204330	1127747108	1166467531	1146388484	1094825953	1118027539
W26	1192416850	1177526497	1151062344	1326221044	1264415084	1359107395	1394363810	1356298162	1289133799	1267370316
W27	1046212273	1035997809	1038048188	1214390939	1168351575	1258896287	1306823569	1265189090	1115138094	1181141678
W28	1129305367	1075158046	1072518594	1267985018	1229994825	1309172601	1306735219	1313875497	1215437565	1203717198
W29	1408551363	1412926523	1403051922	1780100370	1633631923	1717132532	1745610244	1773446798	1533653291	1537637293
W30	1288390742	1319524458	1275627318	1569925222	1493844650	1551742949	1574991911	1586211875	1451255634	1422644739

Tabla 17. Consumo total de energía en kWh de las 10 estrategias durante 30 semanas

Semanas	FFit	Max_u	MaxU_MinC	Min_e	Min_c	Min_ujt	Min_u	MinU_MinC	Rand	RR
W1	208.419425	209.071203	209.093312	274.897905	245.557398	268.422537	280.092751	264.647197	235.698471	242.854222
W2	200.653111	204.26362	198.917917	244.601062	232.180499	242.700903	256.007084	244.867586	235.030679	224.765401
W3	274.277052	272.565722	269.709284	317.576456	312.86034	325.291617	323.957811	322.346	301.830713	296.981109
W4	335.703273	333.413149	331.842404	387.701757	378.999765	406.257681	398.336493	404.859668	368.21278	367.304559
W5	302.413281	297.334821	298.947223	370.212785	352.824772	377.664511	377.47921	377.263061	325.913953	326.267639
W6	253.418801	253.230132	254.297537	307.117144	291.687243	320.818012	319.607585	315.810641	281.333614	283.50627
W7	241.356256	244.509506	241.711472	300.611453	303.845793	318.756138	315.116339	321.988526	270.748278	296.973829
W8	303.99122	306.006542	303.849958	347.143773	343.107701	358.461423	368.745717	362.178412	335.903971	325.301217
W9	322.386931	321.723066	323.122624	358.562245	347.574389	366.002722	358.306746	366.486553	348.016805	343.259649
W10	251.205617	246.319033	249.608663	290.82223	295.90687	301.940609	303.39959	301.052664	274.35159	284.067696
W11	319.348677	318.15131	317.814714	370.594036	358.645752	383.65204	399.401658	396.214197	350.084565	349.048173
W12	295.370672	295.590028	287.084767	356.384161	341.759682	356.510572	364.245214	368.149346	325.298339	335.352447
W13	335.610049	332.874026	324.13333	417.883789	390.203278	415.515031	427.860407	426.141163	363.308875	381.022578
W14	296.119486	295.467769	288.341147	325.576196	311.322846	318.74722	325.033622	316.776046	308.710324	307.87022
W15	342.385047	343.185157	328.475339	372.171359	359.638058	365.976643	375.649185	369.843494	357.689396	357.058799
W16	231.705033	228.878385	228.348135	277.265629	264.294369	288.002757	286.204408	277.886852	253.678846	238.545226
W17	264.993864	262.890205	268.298308	323.808926	282.78736	328.373009	340.423151	314.000597	298.647655	292.208487
W18	231.699793	230.821993	231.282472	256.97849	255.851805	268.241881	264.31821	261.402396	247.975521	244.343765
W19	293.235008	297.16335	290.077581	341.007206	337.831194	349.426181	354.286875	349.624044	317.10988	323.371676
W20	327.210845	326.178819	323.825727	398.431338	371.575951	399.56981	412.226337	402.089912	375.734652	380.285709
W21	318.660684	318.252297	314.44338	440.541137	417.738718	433.793968	442.164808	441.134794	373.560922	377.056164
W22	299.429578	302.671447	302.736875	416.465038	406.826377	427.724842	429.399557	422.234231	374.475969	373.570356
W23	219.52332	221.230834	220.268706	254.658544	238.971859	257.043039	259.939673	255.405898	238.622397	236.024583
W24	267.254697	266.946946	264.117833	304.996433	294.638779	310.395322	303.40547	309.175072	288.086813	297.190671
W25	282.883016	275.485934	272.540482	318.714731	309.223425	313.263086	324.018759	318.441246	304.11832	310.563205
W26	331.226903	327.090694	319.73954	368.394734	351.226412	377.529832	387.323281	376.74949	358.092722	352.04731
W27	290.61452	287.777169	288.346719	337.330816	324.542104	349.693413	363.006547	351.441414	309.760582	328.09491
W28	313.695935	298.655013	297.921832	352.21806	341.665229	363.659056	362.982005	364.965416	337.621546	334.365888
W29	391.264268	392.47959	389.736645	494.472325	453.786645	476.981259	484.891734	492.624111	426.014803	427.12147
W30	357.886317	366.534572	354.340922	436.09034	414.956847	431.039708	437.497753	440.61441	403.126565	395.179094

Tabla 18. Perfil de desempeño de las 10 estrategias

δ	FFit	Max_u	MaxU_MinC	Min_e	Min_c	Min_ujt	Min_u	MinU_MinC	Rand	RR
0	0	0	0	0	0	0	0	0	0	0
0.02	0.766	0.733	0.966	0	0	0	0	0	0	0
0.04	0.933	0.966	1	0	0	0	0	0	0	0
0.06	1	1	1	0	0	0	0	0	0	0.066
0.08	1	1	1	0	0.066	0	0	0	0.1	0.2
0.1	1	1	1	0	0.2	0	0	0.033	0.333	0.333
0.12	1	1	1	0.066	0.266	0.066	0.033	0.033	0.633	0.566
0.14	1	1	1	0.133	0.4	0.1	0.066	0.133	0.866	0.7
0.16	1	1	1	0.233	0.6	0.133	0.166	0.133	0.866	0.766
0.18	1	1	1	0.466	0.766	0.266	0.166	0.266	0.9	0.9
0.2	1	1	1	0.533	0.833	0.3	0.233	0.366	0.966	0.933
0.22	1	1	1	0.6	0.9	0.466	0.4	0.433	0.966	0.933
0.24	1	1	1	0.733	0.9	0.666	0.5	0.6	0.966	0.966
0.26	1	1	1	0.833	0.933	0.733	0.6	0.733	1	1
0.28	1	1	1	0.866	0.933	0.833	0.766	0.833	1	1
0.3	1	1	1	0.9	0.933	0.9	0.833	0.866	1	1
0.32	1	1	1	0.933	0.933	0.9	0.866	0.9	1	1
0.34	1	1	1	0.933	0.966	0.933	0.9	0.933	1	1
0.36	1	1	1	0.933	1	0.933	0.933	0.933	1	1
0.38	1	1	1	0.933	1	0.966	0.933	0.933	1	1
0.4	1	1	1	0.966	1	0.966	0.933	0.933	1	1
0.42	1	1	1	1	1	0.966	0.966	1	1	1
0.44	1	1	1	1	1	1	1	1	1	1

A.4 Resultados del experimento D.

Tabla 19. Consumo total de energía de las 10 estrategias durante 30 semanas

Semanas	FFit	Max_u	MaxU_MinC	Min_e	Min_c	Min_ujt	Min_u	MinU_MinC	Rand	RR
W1	985590956	1030565363	753366596	1052707868	1024634493	1016225902	1066723289	1008125829	973302186	992575563
W2	847116765	911706734	716727730	938384567	866256800	900588068	950458851	908264565	937282189	869913131
W3	1110342596	1107582667	971845335	1180930842	1127327668	1171648816	1225804719	1161386088	1131382184	1133538720
W4	1309256421	1288428890	1195226608	1409598934	1388549947	1476717906	1435252125	1482330075	1388718140	1386718662
W5	1260801822	1231771379	1076793950	1337865588	1287880155	1376747474	1376896220	1376010037	1203415384	1199797473
W6	925864662	924203220	915780072	1109835458	1052729904	1157158539	1151177690	1137229381	1023423773	1021566048
W7	938201620	958345836	870934710	1119815330	1094238488	1148748743	1140759339	1169370387	1104156515	1119528663
W8	1182284686	1206045580	1094498012	1255698291	1240471886	1293988306	1337407811	1308997156	1255209163	1195664918
W9	1411043113	1466496117	1164250939	1383256235	1269052758	1327982183	1318041070	1340689896	1440523861	1368292147
W10	1006491378	1044337456	899253846	1049987926	1092096223	1096019812	1101528300	1092898673	1041027145	1079163599
W11	1395942006	1391946609	1145036168	1371785481	1331592501	1421245501	1484079524	1466688875	1351090362	1332323063
W12	1207297972	1282038208	1034449063	1365787147	1230818816	1291336883	1338764415	1352566894	1204062143	1222603997
W13	1722021051	1787606518	1168059308	1667529524	1676154251	1669451468	1737827551	1729681705	1679343341	1682501013
W14	1541080145	1578911847	1039197565	1308672879	1194116608	1219507065	1300575519	1212374058	1498844465	1445571635
W15	2010800626	1958862480	1184288877	1515658490	1413365872	1434417493	1507747854	1457885759	1589981856	1565340383
W16	899317146	904725220	822724056	1012394850	951816154	1037058578	1033349535	1002658696	928292482	886219172
W17	1026570924	1021398720	966455099	1165866991	1027329967	1185380455	1235269934	1137512786	1106954808	1093119611
W18	910057464	969649067	833307352	943102388	922418399	965973106	955464874	941410298	916735464	896937948
W19	1260072546	1287739540	1045217839	1309081486	1264814356	1306382154	1332251075	1315993675	1248423084	1243012404
W20	1279287753	1297756322	1166393640	1502976234	1349529008	1446381910	1518695223	1456976179	1403761315	1404866918
W21	1474394010	1377712797	1133012902	1557487991	1539750498	1591720592	1628630242	1621825364	1440863723	1465492891
W22	1363911695	1272405972	1090818773	1512051744	1476547958	1551523277	1560461365	1535557451	1455902249	1415007387
W23	884673992	883193539	793711318	912546701	871198301	925825947	938663208	919953388	920377682	873865388
W24	1086933128	1058315413	951733416	1146531255	1061318573	1121171029	1097378801	1113475782	1142037315	1097045117
W25	1341569406	1304035747	982431057	1177460643	1116194284	1129235370	1191786876	1148003931	1179789884	1237377047
W26	1638272072	1597625072	1152376832	1339507623	1281316636	1359672642	1404245447	1358947990	1341756492	1350550791
W27	1295869693	1259502713	1039032071	1258462574	1169015471	1259415216	1331312718	1268748929	1179908916	1219147057
W28	1336452988	1210594147	1073403780	1302278470	1231719341	1309523196	1307208177	1314242119	1288868171	1273864220
W29	1642447267	1648741603	1404493880	1783994822	1635217491	1718129940	1752029695	1774278039	1580346277	1586131715
W30	1877980727	2068308039	1277278159	1789645922	1639587451	1643452363	1706001014	1634796769	1896667169	1665369565

Tabla 20. Consumo total de energía en kWh de las 10 estrategias durante 30 semanas

Semanas	FFit	Max_u	MaxU_MinC	Min_e	Min_c	Min_ujt	Min_u	MinU_MinC	Rand	RR
W1	273.775266	286.268156	209.268499	292.418852	284.620693	282.284973	296.312025	280.034952	270.361718	275.715434
W2	235.310212	253.251871	199.091036	260.66238	240.626889	250.163352	264.016347	252.295713	260.356164	241.642537
W3	308.428499	307.661852	269.957038	328.036345	313.146574	325.458004	340.501311	322.607247	314.272829	314.871867
W4	363.682339	357.896914	332.007391	391.555259	385.708318	410.199418	398.681146	411.758354	385.755039	385.199628
W5	350.222728	342.158716	299.109431	371.62933	357.744488	382.429854	382.471172	382.22501	334.282051	333.277076
W6	257.184628	256.723117	254.383353	308.287627	292.424973	321.432927	319.771581	315.89705	284.284382	283.768347
W7	260.611561	266.207177	241.926308	311.059814	303.955136	319.096873	316.877594	324.825107	306.710143	310.980184
W8	328.412413	335.012661	304.027226	348.805081	344.575524	359.441196	371.50217	363.610321	348.669212	332.129144
W9	391.95642	407.360032	323.403039	384.237843	352.514655	368.88394	366.122519	372.41386	400.145517	380.081152
W10	279.580938	290.093738	249.792735	291.663313	303.360062	304.449948	305.980083	303.582965	289.174207	299.767666
W11	387.761668	386.651836	318.065602	381.051523	369.886806	394.790417	412.244312	407.413576	375.302878	370.08974
W12	335.360548	356.121725	287.346962	379.385319	341.894116	358.70469	371.879004	375.713026	334.461706	339.612221
W13	478.339181	496.557366	324.460919	463.202646	465.598403	463.736519	482.729875	480.46714	466.484261	467.361393
W14	428.077818	438.586624	288.66599	363.520244	331.699058	338.751962	361.270978	336.770572	416.345685	401.547676
W15	558.55573	544.128467	328.969132	421.016247	392.601631	398.449304	418.818848	404.968266	441.661627	434.816773
W16	249.810318	251.312561	228.53446	281.220792	264.393376	288.071827	287.041538	278.516304	257.859023	246.171992
W17	285.15859	283.721867	268.45975	323.851942	285.369435	329.272349	343.130537	315.975774	307.487447	303.644336
W18	252.79374	269.346963	231.474265	261.972885	256.227333	268.325863	265.406909	261.502861	254.64874	249.14943
W19	350.020152	357.705428	290.338289	363.633746	351.337321	362.883932	370.069743	365.553799	346.78419	345.281223
W20	355.357709	360.487867	323.998233	417.493398	374.869169	401.772753	421.859784	404.715605	389.933699	390.240811
W21	409.553892	382.697999	314.725806	432.635553	427.708472	442.144609	452.39729	450.507046	400.239923	407.081359
W22	378.86436	353.446103	303.005215	420.014373	410.152211	430.978688	433.46149	426.543736	404.417291	393.057607
W23	245.742775	245.331539	220.475366	253.485195	241.999528	257.173874	260.73978	255.542608	255.660467	242.740386
W24	301.925869	293.976503	264.370393	318.480904	294.810715	311.436397	304.827445	309.298828	317.232587	304.734755
W25	372.658168	362.232152	272.897516	327.072401	310.053968	313.676492	331.05191	318.889981	327.719412	343.715846
W26	455.075576	443.784742	320.104676	372.085451	355.921288	377.686845	390.06818	377.485553	372.710137	375.152997
W27	359.963804	349.861865	288.62002	349.572937	324.72652	349.83756	369.809088	352.430258	327.752477	338.65196
W28	371.236941	336.276152	298.167717	361.74402	342.144261	363.756443	363.113382	365.067255	358.018936	353.851172
W29	456.235352	457.983779	390.137189	495.554117	454.227081	477.258317	486.674915	492.855011	438.985077	440.592143
W30	521.661313	574.530011	354.799489	497.123867	455.440959	456.514545	473.889171	454.110214	526.851991	462.602657

Tabla 21. Perfil de desempeño de las 10 estrategias

δ	FFit	Max_u	MaxU_MinC	Min_e	Min_c	Min_ujt	Min_u	MinU_MinC	Min_Te	Rand	RR
0	0	0	0	0	0	0	0	0	0	0	0
0.03	0.033	0.033	1	0	0	0	0	0	0	0	0
0.06	0.033	0.066	1	0	0	0	0	0	0	0	0
0.09	0.133	0.1	1	0	0.033	0	0	0	0.133	0	0.066
0.12	0.333	0.3	1	0	0.2	0	0	0	0.233	0.1	0.2
0.15	0.4	0.4	1	0.1	0.4	0.066	0.066	0.033	0.366	0.266	0.266
0.18	0.5	0.533	1	0.2	0.6	0.233	0.1	0.266	0.466	0.5	0.5
0.21	0.566	0.533	1	0.366	0.733	0.3	0.166	0.333	0.633	0.666	0.666
0.24	0.633	0.7	1	0.533	0.8	0.533	0.333	0.5	0.7	0.7	0.7
0.27	0.733	0.733	1	0.633	0.833	0.766	0.5	0.7	0.8	0.733	0.733
0.3	0.733	0.766	1	0.766	0.866	0.833	0.733	0.8	0.8	0.8	0.833
0.33	0.8	0.8	1	0.833	0.866	0.866	0.833	0.833	0.833	0.833	0.933
0.36	0.8	0.8	1	0.833	0.933	0.9	0.866	0.9	0.833	0.9	0.933
0.39	0.833	0.866	1	0.9	0.966	0.9	0.866	0.9	0.9	0.9	0.933
0.42	0.833	0.866	1	0.966	0.966	0.933	0.9	0.933	0.9	0.9	0.966
0.45	0.866	0.866	1	1	1	1	0.966	0.966	0.933	0.966	1
0.48	0.933	0.866	1	1	1	1	0.966	0.966	0.966	0.966	1
0.51	0.966	0.866	1	1	1	1	1	1	0.966	1	1
0.54	0.966	0.933	1	1	1	1	1	1	0.966	1	1
0.57	0.966	0.933	1	1	1	1	1	1	0.966	1	1
0.6	0.966	0.933	1	1	1	1	1	1	0.966	1	1
0.63	0.966	0.966	1	1	1	1	1	1	1	1	1
0.66	0.966	1	1	1	1	1	1	1	1	1	1
0.69	0.966	1	1	1	1	1	1	1	1	1	1
0.72	1	1	1	1	1	1	1	1	1	1	1