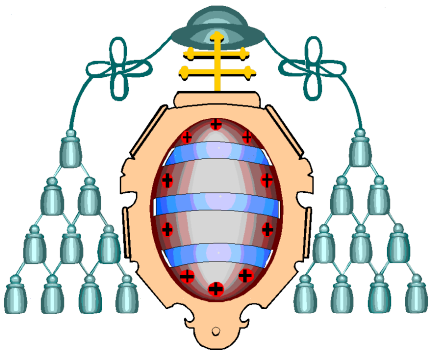

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

ÁREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

TRABAJO DE FIN DE MÁSTER

GESTIÓN AUTONÓMICA DE ENERGÍA Y FALLOS EN CLÚSTERES DE COMPUTADORES

MEMORIA

RAMÓN MEDRANO LLAMAS
ENERO 2012

TUTOR: DANIEL F. GARCÍA
COTUTOR: JOAQUÍN ENTRIALGO

A mis padres.

RESUMEN

El consumo de energía de grandes clústeres de ordenadores afecta a los costes de explotación de centros de datos, en particular incrementando el coste total de posesión (TCO). Por lo tanto, la utilización de algoritmos de minimización de la energía es esencial para una explotación rentable de los centros de datos.

El gestor energético presentado en este trabajo se basa en un algoritmo de provisión dinámica operado de forma autonómica, de forma que se reducen las intervenciones humanas y el sistema es capaz de responder de forma autónoma ante errores, además de operar siempre con un consumo energético mínimo.

Asimismo, la tolerancia a fallos en los grandes centros de datos es fundamental para optimizar el uso de los recursos del mismo y garantizar la consistencia de los datos y del procesamiento que se está llevando a cabo.

En un sistema con provisión dinámica de nodos, donde la topología del clúster es gestionada por un elemento de gestión de carga, es posible aprovechar esta característica para remplazar nodos problemáticos cuando otros estén disponibles.

Este trabajo presenta un sistema de tolerancia a fallos que se integra con el algoritmo de optimización energética basado en provisión dinámica de nodos y permite incrementar la tolerancia a fallos garantizando el cumplimiento del SLA en diversas situaciones de carga y composición del clúster.

ABSTRACT

Power consumption on big clusters is an important component of the operation cost of the data center, in particular, it increases the total cost of ownership (TCO). In this way, the implementation and usage of power optimization algorithms is crucial for an economic-wise management of the data center.

The power manager introduced in this work is based on a dynamic provisioning algorithm autonomically managed that reduces human interactions and makes the system able to run using less energy and respond to errors automatically, as well.

In addition, fault tolerance on big data centers is essential for the optimization of its usage and the warranty the coherence and consistency of the data al the process made on that data center.

In a cluster whose nodes are dynamically provisioned, and which its topology is managed by a load balancing element, it is possible to take advantage of this feature to replace nodes with issues when free nodes are available.

This work presents a fault tolerance system that integrates with the power optimization algorithm based on dynamic provisioning of nodes and enables the improvement of the fault tolerance while guaranteeing the fulfillment of the SLA on multiple load scenarios and cluster configurations.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE GRÁFICOS

1 INTRODUCCIÓN

Este trabajo fin de máster se trata de una continuación del trabajo realizado en el Proyecto Fin de Carrera de Ingeniería Informática realizado en la Escuela Politécnica de Ingeniería de Gijón de la Universidad de Oviedo y titulado "Gestión energética basada en métricas de calidad de servicio" (numerado 1102015).En dicho proyecto se establecieron las bases de un algoritmo de auto-optimización del consumo energético para clústeres escalables que permite, en las pruebas realizadas, reducir el consumo energético de un clúster entre un 17% y un 85% mediante la técnica de provisión dinámica de nodos de forma autonómica.

La computación autonómica fue definida en [Kephart03] de forma derivada del manifiesto producido por IBM en [Horn01]; de esta forma, se define como "sistemas de computación que pueden administrarse a sí mismos dados una serie de objetivos de alto nivel por parte del administrador".

Tal y como se puede deducir de esta definición, obtener sistemas capaces de auto administrarse, lo que incluye configurarse, adaptarse y repararse, es prácticamente un sueño a día de hoy y sólo puede ser conseguido incrementado poco a poco el nivel de abstracción de la manipulación del sistema y evolucionando los sistemas y modelos existentes.

Fundamentalmente, este trabajo de fin de máster está construido sobre el proyecto fin de carrera, y está centrado en la extensión en relación de la tolerancia a fallos del algoritmo de optimización energética original.

La tolerancia a fallos en clústeres que proveen servicios para Internet no es un problema simple, pues requiere de técnicas avanzadas en sistemas distribuidos para coordinar transacciones o recuperar sesiones de forma fiable, eficiente y, sobre todo, coherente con los datos y los clientes.

Los clústeres de escalabilidad o balanceo de carga son una herramienta muy flexible de cara a incrementar la capacidad de un servicio de forma horizontal, sin embargo el coste energético de estos sistemas supone un reto cada día más grande para la amortización de un sistema, motivo por el cual se introducen sistemas de optimización de energía.

Este tipo de clústeres es, además, un caso especial de la tolerancia a fallos, donde, al tener muchos elementos funcionando en paralelo, la tasa de fallos del sistema completo se reduce notablemente. Es por esto que una evolución de la gestión autonómica debe incluir, además de la optimización energética, la maximización de la fiabilidad del servicio proporcionada a los clientes.

La investigación aquí presentada se centra en la corrección y optimización del gestor energético en entornos de clúster heterogéneos y la extensión de sus capacidades autonómicas a la tolerancia a fallos.

Este trabajo se ha realizado de forma incremental sobre el proyecto que presentaba el gestor energético, en primer lugar detectando, corrigiendo y validando los problemas relativos a su uso en clústeres heterogéneos y en segundo lugar diseñando, implementando y validando mediante los experimentos que aquí se presentan el módulo encargado de gestionar los fallos de los nodos que se integra con el gestor energético.

El documento se estructura como sigue: en primer lugar se presenta un resumen del estado del arte en relación a técnicas de gestión energética autonómicas y su evolución en los últimos años además de técnicas comunes de tolerancia a fallos en computación basada en clúster. A continuación se presenta una introducción al gestor energético, así como el análisis a los problemas encontrados para clústeres heterogéneos, seguida de la descripción y diseño del módulo de gestión de tolerancia a fallos. El capítulo siguiente está dedicado al diseño experimental y a las pruebas de validación realizadas. Finalmente, el último capítulo queda reservado al análisis de las conclusiones y el trabajo futuro.

2 ESTADO DEL ARTE

La hoja de ruta relativa a la gestión energética fue establecida en [Pinheiro01]. Tras observar las diferentes líneas de investigación, que incluyen técnicas como provisión dinámica de nodos, gestión de frecuencias o de máquinas virtuales; y desarrollar el algoritmo de gestión energética que se basa en métricas de calidad de servicio (QoS) para realizar la optimización, el siguiente paso se centra en la integración de técnicas avanzadas de tolerancia a fallos y control de admisión en el algoritmo de gestión, con el objetivo de ofrecer una solución completa de auto optimización (*self-optimization*).

La revisión al estado del arte se ha estructurado en torno a los tres ejes fundamentales que permiten diseñar un sistema de gestión autonómico: la optimización energética, el control de admisión para la garantía del cumplimiento del SLA y el balanceo de carga para su aplicación en clústeres de computadores.

2.1 Evolución de técnicas de gestión de energía

La gestión de energía para computadores en clúster ha evolucionado desde una gestión individual y basada en rangos simples para variables como el tiempo de inactividad a modelos complejos con sistemas de optimización basados en algoritmos de diversas clases que tienen en cuenta métricas tomadas en tiempo real de colectores de métricas de múltiples niveles en una jerarquía de computadores.

2.1.1 Gestión no autonómica

La gestión no autonómica se caracteriza por tratarse de una gestión puramente centrada a nivel del nodo que se está manejando, esto es, podría describirse también como una gestión

energética no integrada a nivel de clúster y centrada en sistemas que optimizan cada uno de los nodos de forma local.

El ámbito de aplicación de las acciones realizables por el actuador del gestor energético local se reduce asimismo al computador local, pudiendo ir éstas desde un nivel de granularidad más fino, como reducir la velocidad de giro de los discos duros a otras de un nivel más grueso, como poner a todo el computador en un estado de hibernación.

Normalmente, los sensores tienen en cuenta únicamente métricas locales, tales como la utilización de la CPU o el tiempo de inactividad del disco y sólo se permite aplicar políticas basadas en rangos simples que no tienen en cuenta su historial [Yung00].

Sin embargo, la simplicidad de estos modelos de gestión no resta eficacia en determinados ámbitos; por ejemplo en computadores de escritorio y portátiles, donde el tiempo de inactividad es muy amplio, el control con ajuste dinámico de frecuencia (DVFS) se ha mostrado muy útil para reducir el consumo del procesador y, además de prolongar la vida de las baterías en computadores portátiles, reducir el ruido producido por sistemas de disipación.

Existen alternativas para medir el consumo energético de un clúster, de forma que tomando medidas para diversas configuraciones es posible seleccionar la que presenta el menor consumo.

Para esas medidas es posible usar benchmarks específicos, tales como el descrito en [WangX08], que define agentes de medida para cada uno de los nodos del clúster. Existen alternativas más avanzadas, como los algoritmos de programación dinámica mostrados en [ChenJ11].

2.1.2 Técnicas basadas en provisión dinámica de nodos

La provisión dinámica de nodos se centra en la configuración de la topología del clúster de forma que, conforme la carga del servicio evoluciona, esta topología se adapta para proporcionar la cantidad de nodos encendidos en cada momento que permitan satisfacer los requisitos de calidad del servicio minimizando el número de nodos en funcionamiento.

Una característica deseable de este tipo de técnicas es la gestión autonómica, de forma que el actuador aplica las modificaciones a la topología de forma automática conforme los sensores detectan los cambios de la carga soportada.

En esta técnica es capital poder predecir el comportamiento del clúster dada una carga de usuarios y, además, si es posible, poder predecir la forma de la carga a soportar en base al historial de carga que se tiene.

Tal y como se puede deducir, se trata de una técnica de gestión que incrementa el nivel de abstracción para tener una visión global del clúster que está gestionando, además de necesitar un modelado detallado tanto de la carga como del clúster.

Por ejemplo, la estrategia mostrada en [Rusu06] ofrece provisión dinámica de nodos en base a su carga individual, calculando cuáles forman el conjunto mínimo de nodos necesario para soportar la carga solicitada. Cada nodo se caracteriza por la proporción de carga del clúster que puede soportar, y no por medidas de eficiencia energética.

Es necesario analizar el impacto de la provisión dinámica de nodos, bien mediante técnicas experimentales o bien mediante análisis de colas para estimar cómo puede afectar al rendimiento del servicio, al consumo energético y de qué parámetros depende. En [Guerra08] se realiza un repaso a diversas opciones de análisis.

El objetivo principal de la provisión dinámica, además de la concentración de carga para reducir el número de nodos encendidos, puede venir complementado por otros objetivos adicionales, tales como el apagado de nodos en función de la temperatura para reducir costes de refrigeración [Ramos08].

2.1.3 Técnicas con DVFS integrado

Las técnicas DVFS (*Dynamic Voltage and Frequency Scaling*) permiten alterar la frecuencia de reloj del procesador, así como su voltaje de funcionamiento [Nowka08].

Existen técnicas de autogestión de energía que permiten manejar de forma centralizada el nivel de frecuencia de los procesadores del clúster que están gestionando. Adicionalmente, suelen integrarse con sistemas de provisión dinámica de nodos, permitiendo el desarrollo de sistemas de autogestión más flexibles [Kim09].

Este mayor nivel de flexibilidad lleva consigo un incremento de la complejidad de los modelos utilizados para describir al servicio, el clúster y la carga esperada y también de la cantidad de parámetros que rigen estos modelos, haciendo que su ajuste y adaptación a cada servicio sea más difícil y costosa. Además, el gestionar de forma explícita la frecuencia sólo incrementa la

eficiencia en una pequeña proporción, mientras que dejar esta gestión a nivel del sistema operativo proporciona un muy buen compromiso entre complejidad y consumo energético [Yum02].

En general, en computadores dedicados a la ejecución de servicios para Internet, la ganancia que se obtiene, en términos de ahorro energético, es mayor mediante la provisión dinámica de nodos que mediante la gestión de DVFS.

2.1.4 Provisión dinámica de máquinas virtuales

Uno de los efectos que producen los sistemas de autogestión de energía que realizan provisión dinámica de nodos es la tendencia a la concentración de la carga en el número mínimo de nodos que permiten ofrecer el servicio dentro de los parámetros de calidad de servicio establecidos [Woods07][Murphy09].

Se concentra la carga en un número menor de máquinas debido a que, en general, éstas son más eficientes en rangos de carga alta, donde el consumo de los dispositivos que no poseen sistemas de gestión de energía proporcional a la carga que soportan (tales como los discos duros, que sólo se apagan cuando no están soportando carga alguna, o los módulos de memoria, que necesitan refrescarse permanentemente) están soportando un nivel de carga más alto, de forma que la métrica de energía por usuario tiende hacia valores de mayor eficiencia [Kim09].

La consolidación de la carga es un problema altamente dependiente del tipo de servicio que se esté optimizando, de forma que un servicio de comercio electrónico requerirá de mecanismos de migración de sesiones, mientras que un servicio de base de datos distribuida requerirá mecanismos de recuperación de transacciones y de coordinación de datos [Wang11a].

Esta consolidación de la carga se realiza de forma más sencilla y genérica mediante la distribución de máquinas virtuales, que son relativamente más fáciles de migrar de una máquina física a otra y permiten abstraerse del servicio que están proporcionando.

Un problema relacionado con la distribución de máquinas virtuales es la sobrecarga que genera y los tiempos de espera para la clonación de la máquina virtual. Estos factores hacen más complejo garantizar el cumplimiento del SLA (Service Level Agreement, acuerdo de nivel de servicio) debido a la menor predictibilidad de estos tiempos [Kusic09].

2.1.5 Gestión energética autonómica jerárquica

La gestión jerárquica o multinivel considera una integración de todos los mecanismos anteriores de forma centralizada, integrando gestión a nivel de clúster con, por ejemplo, provisión dinámica de nodos, permitiendo la migración de carga de forma adecuada al servicio y con una gestión a nivel de nodo, estableciendo políticas locales de gestión energética para el control de los dispositivos mediante rangos de inactividad y gestión local del DVFS [Wang11b].

La gestión multinivel es muy flexible, de forma que en cada nivel pueden aplicarse diferentes técnicas de control. Por ejemplo, en [Medrano11] se estableció un sistema autonómico de provisión dinámica de nodos, mientras que la gestión a nivel de nodo se realizó con el propio sistema de gestión energética del sistema operativo.

Estas técnicas de gestión energética pueden integrarse en sistemas de autogestión, que cuentan con una visión global del estado del clúster y son capaces de tomar decisiones de forma autónoma.

La gestión de cada uno de los niveles puede estar más o menos aislada de la de los niveles superiores, de forma que cada capa de optimización puede proporcionar información a las capas superiores para tener una visión más completa.

El conectar las capas, sin embargo, incrementa la complejidad del algoritmo, además de presentar mayores problemas de integración con sistemas existentes al tener mayor acoplamiento con todos los niveles.

2.1.6 Integración de requisitos de *QoS*

De cara a garantizar los requisitos de calidad de servicio, es fundamental que el gestor de autonómico de energía conozca en todo momento la situación en la que se encuentra el clúster, el nivel de carga, la utilización de dispositivos y la fracción de capacidad que se está utilizando.

Estas métricas pueden variar dinámicamente [Garcia09] y, en función de ellas, el gestor energético tiene que operar de una forma u otra.

En general, sin la posibilidad de recolectar este tipo de métricas no es posible realizar una completa optimización del servicio ya que se desconoce la situación instantánea del servicio.

Existen algunos algoritmos que utilizan métricas de calidad de servicio obtenidas del clúster, bien de forma dinámica o mediante un experimento de modelado inicial [Weiser94].

No es posible garantizar el cumplimiento de las restricciones de calidad de servicio establecidas por los acuerdos de nivel de servicio (SLA) si no se conoce el comportamiento del clúster en condiciones reales de carga [Lu01]; además, es necesario utilizar métricas medibles de forma directa y que sean comparables con los requisitos expresados en el SLA.

La utilización de métricas como la productividad (*throughput*) instantánea o la cantidad de clientes soportados no tiene por qué implicar que se esté cumpliendo con los requisitos de servicio para cada uno de los clientes. Por esto, métricas como el percentil-90 del tiempo de respuesta, latencia o tiempos de acceso son más apropiadas para modelar la calidad de la respuesta temporal de los servicios y mantenerlos monitorizados.

Existen trabajos relativos a provisión dinámica de nodos que o bien no utilizan métricas de calidad de servicio de ningún tipo [Bertini07][Bertini10a][Bertini10b], o bien utilizan métricas que no se traducen en una medida directa de la calidad de servicio proporcionada a los clientes, tales como productividad o uso de recursos [Elnozahy03a][Elnozahy03b].

Finalmente, sólo el trabajo de [ChenY05] tiene en cuenta de forma explícita el *SLA* del servicio. El trabajo de [Xue07], de provisión dinámica de nodos incluye ciertas restricciones sobre la calidad del servicio, pero ninguna garantía.

2.1.7 Gestión autonómica integrada

Finalmente, el último paso en la gestión autonómica es realizar un gestor que permita una auto-optimización completa del servicio (*self-optimization*) que incluya la optimización energética, tolerancia a fallos y balanceo de carga en un mismo sistema de optimización.

Los sistemas que incluyen estos gestores son normalmente más complejos e integran múltiples orígenes de métricas y modelos [Lefurgy03], de forma que permiten ofrecer mucha más flexibilidad para la gestión de la energía del clúster utilizando dichas métricas adecuadas a cada nivel de optimización que se esté realizando. Por ejemplo, se pueden usar métricas como la temperatura de la CPU para la gestión local del DVFS de un nodo, mientras que la gestión de energía a nivel más alto puede requerir métricas como el tiempo de respuesta instantáneo.

El cumplimiento del SLA es muy importante y es necesario garantizarlo. En [Wang10] se propone un mecanismo interesante que permite medir la probabilidad de violación del mismo en función de diversos parámetros, como la relación de cercanía entre el blade y el ventilador más cercano, la capacidad total de refrigeración y las limitaciones energéticas. Además, se permite establecer la probabilidad de violar el presupuesto energético, a costa de incrementar la posibilidad de violar el SLA. Principalmente, el problema de gestionar todos estos factores de forma centralizada es la alta complejidad del modelo necesario y la dificultad de optimizar un número elevado de parámetros.

2.2 Técnicas de balanceo de carga y control de admisión

Existen diversas técnicas de balanceo de carga y control de admisión para clústeres. A continuación se analizan las diversas técnicas y sus posibilidades de integración con mecanismos de gestión autonómica [Fox97].

2.2.1 Control de admisión y garantía de calidad de servicio

El control de admisión es, junto con la recolección de mediciones de QoS en tiempo real, la otra técnica fundamental para garantizar la calidad de servicio. De hecho, se basa en la primera para tomar decisiones.

Los mecanismos de control de admisión permiten garantizar que el servicio no va a estar siendo usado por más clientes que los soportados. Esta cantidad de clientes puede ser determinada en base a restricciones de capacidad del hardware o a restricciones de tiempo de respuesta, mediante el uso de modelos descriptivos del clúster [Garcia10].

Teniendo en cuenta los datos obtenidos de los recolectores de mediciones de calidad de servicio, es posible conocer la capacidad máxima del clúster para el servicio y comprobar en qué nivel de utilización se encuentra. De esta forma, además del control de admisión, se pueden tomar decisiones más avanzadas, como tender a la concentración la de carga mediante el apagado de nodos.

En este sentido, es fundamental conocer de forma precisa el comportamiento del clúster, por lo que hay que haberlo modelado previamente. Además, hay que utilizar métricas adecuadas.

En este caso, usar métricas de utilización sí puede ser apropiado, pero es conveniente ser capaz de predecir cuál es la capacidad máxima del servicio y cuáles son los requisitos de recursos por cliente.

En [Cherkasova02] se propone un mecanismo basado estrictamente en métricas de utilización, que permite garantizar que un servidor web no se sature.

La saturación de un servidor o clúster es un problema que perjudica la calidad de servicio de todos los clientes, incrementando los tiempos de respuesta y, potencialmente, perdiendo oportunidades de venta, lo que perjudica la reputación de un servicio.

El control de admisión viene a resolver este problema mediante el establecimiento de límites en la cantidad de clientes que un servidor puede soportar para un determinado nivel de calidad de servicio. Por encima de la capacidad máxima, no se admiten más clientes [Elnikety04].

En este caso, el control de admisión no debe sólo ser capaz de establecer límites en la capacidad del servicio, sino que además debe de identificar tipos de clientes y reservar recursos para cada uno de ellos adecuadamente. De esta forma es muy importante conocer el comportamiento esperado del clúster. En el estado del arte estudiado, ningún sistema utiliza modelos de calidad de servicio para el control de admisión, lo cual es fundamental

2.2.2 Balanceo de carga en clústeres

El balanceo de carga en clústeres es uno de los objetivos principales de la computación en clúster. Un servicio que permita de forma sencilla repartir la carga entre diversas computadoras permite incrementar la capacidad hasta en un factor n, siendo n el número de nodos trabajadores (que no la velocidad individual de la respuesta, de ahí que se construyan servicios de alta productividad) [Dean08].

Normalmente, en clústeres de balanceo de carga, se reparte el trabajo de forma proporcional a la capacidad de cada nodo trabajador para tener un tiempo de respuesta lo más uniforme posible.

Existen diversas técnicas para el balanceo de carga, desde las basadas en mecanismos de rotación de recursos de red [Aversa00], como las direcciones de red asignadas a nombres de dominio, direcciones IP virtuales o soluciones más específicas, y flexibles, como el entunelamiento de paquetes mediante un nodo de reparto de carga [Candellini99].

El balanceo de carga está íntimamente relacionado con la tolerancia a fallos, tal y como se explica en la sección 2.3. En servicios fácilmente replicables por capas, como servicios web, una distribución de carga ayuda en gran medida a incrementar la tolerancia a fallos [Narendran97].

Fundamentalmente, pueden considerarse dos tipos de técnicas de balanceo de carga. En primer lugar, se definen las técnicas centralizadas, donde un nodo dentro del clúster tiene la responsabilidad especial de recibir las peticiones de los clientes externos y, en función de determinadas políticas, distribuir el procesamiento de la carga entre los diversos nodos trabajadores del clúster de forma que se equilibre la carga entre ellos y se consiga aprovechar toda la capacidad.

La segunda categoría de técnicas son, por el contrario, no centralizadas. Con estas técnicas, se tiene un conjunto de nodos que se coordinan de igual a igual para repartirse la carga sin necesidad de elementos adicionales.

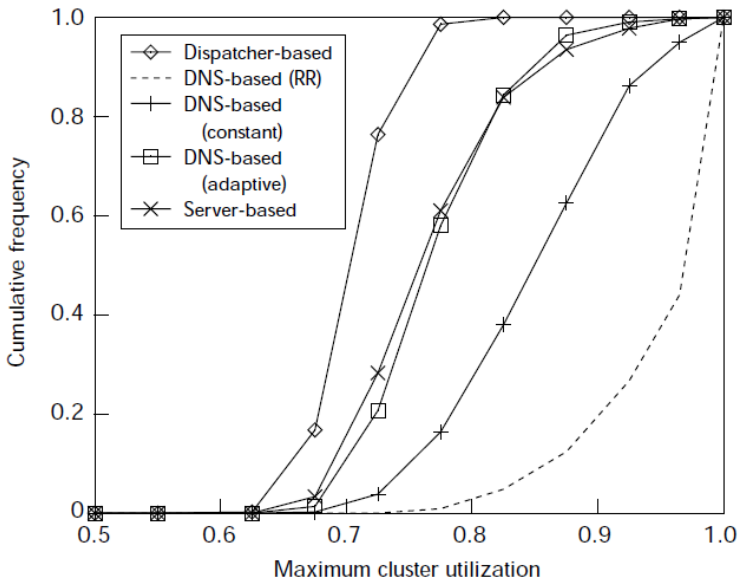

Gráfico 1: Utilización esperada del clúster con diversos mecanismos de balanceo de carga[1]

Dependiendo del mecanismo de balanceo de carga utilizado, la utilización esperada del clúster variará y realizar un análisis detallado de la misma es fundamental, tal y como se muestra en el Gráfico 1, que relaciona la distribución acumulada de la utilización máxima del clúster en relación a diversas técnicas de balanceo de carga. En este caso, el mecanismo de distribución de carga que permite alcanzar una utilización media mayor del clúster es el basado en DNS round-robin, tal y como se demuestra al observar como se acumula mucha más probabilidad en la parte

[1] Figura obtenida de [Candellini99].

alta del gráfico (utilizaciones cercanas a 1.0). Este cálculo de probabilidad se ha realizado con múltiples experimentos de carga, de forma que el gráfico representa la probabilidad acumulada de encontrar una utilización máxima de este nivel.

Para implementar el mecanismo de balanceo, además de la distribución de carga basada en los niveles de utilización de cada nodo, pueden usarse otras variables complementarias, como el consumo de cada máquina y el coste energético de cada sesión [WangX08].

Una posibilidad para la evaluación de diferentes estrategias de balanceo de carga se explora en [Yong01], donde mediante el análisis de trazas de funcionamiento de un servicio se comparan estrategias basadas en DNS, donde no es necesario un nodo especial que distribuya la carga entre los diferentes nodos de trabajo, con la alternativa basada en un nodo especializado.

El balanceo de carga a nivel de red es potencialmente sencillo de integrar en servicios existentes, pero generalmente presenta menos flexibilidad que las técnicas basadas en un nodo especializado de repartición de carga.

Los objetivos del balanceo de carga, además de mantener niveles de utilización igualitarios entre los diversos nodos o de concentrar la carga en el mínimo número necesario, pueden ser más variados y dependientes del modelo de servicio que se esté optimizando, por ejemplo, para nivelar la terminación de las tareas en un mecanismo de división de procesos [Bevilacqua99].

2.2.3 Integración del control de admisión y balanceo de carga

En general el balanceo de carga y el control de admisión pueden realizarse en el mismo proceso, seleccionando el nodo trabajador al que asignar la nueva sesión de forma que se nivela la carga entre todos los nodos y se toma la decisión de aceptar o no la nueva sesión.

Además, en los momentos de toma de decisiones que influyen en el comportamiento de la calidad del servicio ofrecido, también puede realizarse la provisión dinámica de nodos.

Existen diversas aproximaciones a una estrategia de control de admisión y balanceo de carga integrada, tales como el establecimiento de una cola de solicitudes en un nodo distribuidor [Sharifiana08] o el uso de ratios de aceptabilidad para el control de admisión y proporciones de carga por nodo descritas en [Aweya02].

2.3 Tolerancia a fallos

A continuación se analizan diversos mecanismos de tolerancia a fallos aplicables a servicios basados en clústeres.

2.3.1 Detección de fallos

La detección de fallos en sistemas distribuidos puede ser prematura o realizarse tras el fallo. En la sección 4.1.1 se abunda en detalles para la implementación de ambas.

Una estrategia muy común de detección de fallos es el uso de sistemas de latido (*heartbeat*) que permiten monitorizar los estados de los nodos trabajadores y conocer cuándo se producen los errores. Estas estrategias se basan en que cada nodo trabajador irá notificando de forma regular su disponibilidad al nodo controlador, de forma que cuando éste no recibe la notificación de uno de los nodos trabajadores, se determina que dicho nodo ha fallado. El principal problema de estas estrategias es que suponen un coste de despliegue de las notificaciones y de uso de recursos de red y de procesamiento de las notificaciones [Kalbarczyk99].

2.3.2 Recuperación y migración de sesiones

Si se denomina sesión a la unidad de procesamiento de un servicio transaccional (podría ser, por ejemplo, un conjunto de transacciones de un servicio de bases de datos), la recuperación de sesiones consiste en rehacer el trabajo que se ha perdido cuando uno de los nodos trabajadores del clúster de balanceo se ha caído.

La recuperación de sesiones incrementa el tiempo de respuesta en caso de caída de uno de los nodos trabajadores y, además, introduce picos en los niveles de carga cuando se está produciendo la migración de las sesiones de los nodos caídos, incrementando la probabilidad de que se produzcan períodos de tiempo donde se viole la calidad de servicio.

La recuperación de carga requiere una adecuada gestión de la redundancia, de forma que aparecen nuevos retos tales como la coordinación de transacciones o la recuperación de trabajos [Petri95].

En el ámbito de la gestión energética aparecen otros interrogantes, como por ejemplo, los relativos a qué nodo asignar las sesiones ya admitidas cuando uno de ellos falla, teniendo en cuenta el consumo energético, y cómo afecta eso a la calidad de servicio percibida por el cliente.

La diferencia entre una estrategia de recuperación de sesiones y una estrategia de migración de sesiones radica en que la recuperación de sesiones repite el trabajo ya realizado y la migración continúa la ejecución de la sesión en otro nodo trabajador [Kandaswamy08].

Un objetivo de la investigación en la migración de sesiones es evolucionarla hacia un modelo más transparente, lo que permitiría desarrollar mecanismos de migración más genéricos y adaptables a diversos modelos de sistemas de forma sencilla. Uno de estos mecanismos se basa en hacer migraciones de las conexiones de red en el nivel de internet [Vishnu09].

La migración de sesiones puede realizarse asimismo de forma proactiva usando técnicas de aprendizaje por computador y computación autonómica para reducir el tiempo de recuperación, con el necesario incremento de los recursos necesarios para llevar a cabo esta recuperación proactiva [Nagarajan07].

2.3.3 Checkpointing

La técnica del checkpointing consiste en almacenar resultados intermedios del procesamiento de una transacción con el objetivo de, que si falla el nodo de trabajo que está realizando este procesamiento, rehacerlo en otro de los nodos de trabajo, pero no totalmente.

El checkpointing se implementa de forma general en sistemas de alta productividad, como los sistemas de ficheros distribuidos y de procesamiento como las implementaciones de Hadoop [Borthakur07] o Google, GFS y MapReduce [Dean08].

Esta técnica es relativamente dependiente del sistema en donde está implementada, de forma que en cada caso se ejecutarán acciones diferentes. No obstante, hay opciones de más bajo nivel que permiten hacer checkpointing a procesos de forma directa, además de su migración a otras máquinas, aunque todas han de tener la misma arquitectura, no soportando clústeres heterogéneos [Kalbarczyk99].

La principal desventaja del checkpointing es el tiempo de recuperación y la gran cantidad de recursos que requiere para el logging de la información que permita rehacer el trabajo una vez perdido uno de los nodos trabajadores del clúster [Ropars11].

2.3.4 Paralelismo y sobre-previsión de recursos

Otro mecanismo para no cortar el servicio a los clientes es la sobre-previsión de recursos, replicando siempre todas y cada una de las peticiones que se realizan y devolviendo una sola de las respuestas.

Esto soluciona los problemas del modelo de recuperación (podría llamarse un modelo proactivo, en lugar de recuperación), sin embargo hace que el servicio requiera multiplicar la capacidad por el ratio de paralelismo seleccionado para soportar la misma cantidad de usuarios, lo cual no es económicamente viable ni energéticamente eficiente salvo en situaciones de extrema criticidad del tiempo de respuesta.

En general, en una estrategia de optimización energética, no tiene sentido aplicar mecanismos agresivos de sobre-previsión puesto que la eficiencia energética de cada cliente se divide por el número de réplicas de procesamiento que se lancen. Además, se presentan problemas de coherencia de datos.

Este paralelismo puede ser aplicado a cualquier nivel de la pila de software, como a procesos, servidores web o sistemas de base de datos [Shye09]

2.3.5 Integración de tolerancia a fallos en software existente

Uno de los objetivos deseables de una estrategia de tolerancia a fallos es aplicarla a sistemas software existentes de forma rápida y con bajo coste de integración. Por esto, es muy conveniente que la implementación de estas estrategias venga en forma de librerías fácilmente enlazables o middleware sobre el que construir nuevos sistemas o integrar sistemas existentes.

Esta segunda forma de distribución del software es relativamente flexible para la construcción de software, pero no para la integración en software ya desplegado [Wagle11].

La integración con sistemas existentes mediante tecnologías de introspección [James09] o librerías con un mínimo de puntos de integración es más flexible y maximiza las posibilidades de despliegue de políticas autonómicas.

2.4 Conclusiones al análisis

Fundamentalmente, la conclusión del análisis del estado del arte es la no existencia de mecanismos integrados (*self-optimization*) de balanceo de carga, control de admisión y gestión energética con tolerancia a fallos que permitan gestionar de forma autonómica un gran clúster de computadores.

Adicionalmente, los modelos en los que se basan los mecanismos existentes no son lo suficientemente sencillos como para ser integrados en sistemas ya desplegados para proporcionar los requisitos anteriores de forma integrada.

Si bien existen sistemas que combinan balanceo de carga y control de admisión [Cherkasova02] [Sharifiana08], ninguno tiene en cuenta el consumo energético (*power-aware*) para estas tareas y, además, no se utilizan métricas apropiadas para la calidad de servicio en la gestión de control de admisión, sino métricas de tipo interno, como la utilización de CPU o el tiempo de latencia de disco.

2.4.1 Aportación al estado del arte

El enfoque propuesto en este trabajo integra los elementos fundamentales de un gestor autonómico con el objetivo de la optimización energética garantizando la calidad de servicio ofrecida a los clientes: el control de admisión, la gestión energética basada en provisión dinámica de nodos y un sistema de tolerancia a fallos adaptado a los elementos anteriores.

Para ello se utilizan exclusivamente modelos sencillos del clúster que proporcionan métricas estrictas de calidad de servicio y que permiten garantizar en todo momento el cumplimiento de la calidad de servicio, integrando mecanismos de balanceo de carga en clústeres escalables, control de admisión centralizada, tolerancia a fallos mediante recuperación de sesiones y gestión energética de forma autonómica.

Tal y como se ha observado anteriormente, una solución integrada y adaptable a cualquier software existente es fundamental para incrementar el grado de implantación de sistemas de computación autonómica y mejorar la calidad de servicio mostrando un menor número de fallos a los usuarios, y eso es lo que consigue el trabajo aquí presentado.

2.4.2 Comparación de este trabajo con el estado del arte

En la Tabla 1 se resumen los principales aspectos de los trabajos analizados en el estado del arte que tienen más relación con el trabajo realizado.

La tabla está organizada de forma que en cada fila se presenta un trabajo y en las columnas se establecen las características de cada uno, marcando un cuadro relleno las que corresponden y con un cuadro vacío las que no. Como se puede observar, no hay ningún trabajo que cubra exactamente los puntos tratados en este trabajo de fin de máster.

	Gestión energética					**Balanceo**			**Adm.**		**R. Errores**		
	Provisión dinámica	Gestión integrada de DVFS	Gestión local de DVFS	Garantía del SLA	Modelos de respuesta	Balanceo por round-robin simple	Balanceo adaptativo	Balanceo adaptativo basando en QoS	Admisión basada en políticas simples	Admisión basada en modelos de respuesta	Migración de sesiones	Checkpointing	Sobreprovisión/paralelismo
[Aversa00]						■			■				
[Aweya02]							■		■				
[Bertini07]	■					■			■				
[Bertini10a]	■	■					■		■				
[Bertini10b]	■	■				■			■				
[ChenJ11]	■			■	■		■		■				
[Cherkasova02]						■			■				
[Dean08]						■						■	■
[Elnozahy03a]		■											
[Elnozahy03b]		■											
[Kalbarczyk99]											■	■	
[Narendran97]							■						
[Petri95]						■					■		
[Pinheiro01]	■					■			■				
[Rusu06]	■	■					■		■				
[Wang11a]	■			■			■		■				
[Wang11b]	■					■			■				
[WangX08]	■				■				■				
Este trabajo	■		■	■	■			■	■	■	■		

Tabla 1: Trabajo relacionado

Fundamentalmente, pueden destacarse dos grupos de trabajos: los centrados en la gestión energética y los centrados en tolerancia a fallos. Ninguno de los trabajos analizados propone un enfoque que integre los tres componentes anteriormente explicados de gestión energética, control de admisión y balanceo de carga para la garantía del SLA y tolerancia a fallos.

En cuanto a la gestión energética, este trabajo es el único que presenta un enfoque de provisión dinámica de nodos que, garantizando el cumplimiento del SLA, utiliza modelos de respuesta medidos de forma empírica para la optimización energética. Asimismo, estos modelos permiten realizar un balanceo de carga inteligente, pero basado en modelos de respuesta que aseguran el cumplimiento del SLA en todos los nodos. Los trabajos relacionados analizados se centran en utilizar políticas y modelos sencillos, tales como la carga media de las máquinas. Por último, en el apartado de tolerancia a fallos, se ha decidido utilizar un modelo de migración de sesiones dado lo eficaz que es y la característica de no necesitar de recursos adicionales en todo momento para la tolerancia a fallos como los demás mecanismos citados.

En el ámbito de la gestión energética, los trabajos de [Bertini07] y [Bertini10a], así como el de [Rusu06] son los que presentan una mayor integración con mecanismos de balanceo de carga adaptativo, pero no garantizan el cumplimiento del SLA. Son los trabajos de [ChenJ11] y [Wang11a] los que sí garantizan el cumplimento del SLA, con el coste de utilizar modelos complejos y difíciles de optimizar. En cuanto a la tolerancia y recuperación de fallos, son los trabajos de [Dean08] y [Kalbarczyk99] los que presentan enfoques más avanzados, sin embargo, ninguno de ellos realiza gestión energética o control de admisión.

3 GESTOR DE ENERGÍA

En esta sección se presenta un resumen introductorio al algoritmo de gestión energética diseñado y validado en [Medrano11]. Este algoritmo se ha integrado en un prototipo de servicio transaccional que permite su validación y prueba.

En este capítulo se hace una introducción al algoritmo que ejecuta el gestor energético y se describen las mejoras para su integración en clústeres heterogéneos. En el Capítulo 4 se describe en detalle el gestor de tolerancia a fallos.

3.1 El algoritmo de optimización

El algoritmo diseñado y prototipado se ejecuta en el nodo distribuidor de forma centralizada, tal y como se puede observar en el Gráfico 2, y tiene como objetivo la toma de decisiones en dos vertientes: asistir al control de admisión informando sobre la capacidad del clúster en cada momento y tomar decisiones de control energético, esto es, apagar o encender nodos para satisfacer el SLA del servicio.

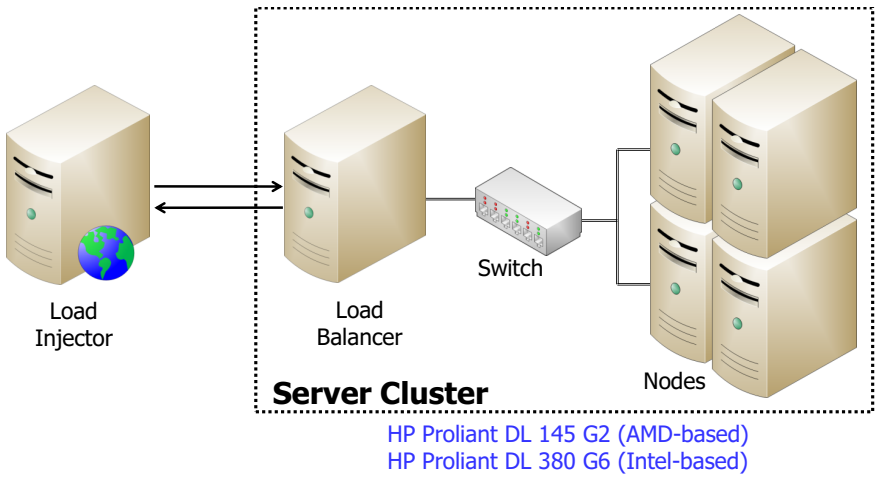

Gráfico 2: Diagrama del clúster a optimizar

Para tomar las decisiones, el algoritmo necesita tener conocimiento sobre el comportamiento del tiempo de respuesta de cada nodo i, denotado por R_i, en función del número de sesiones activas, denotado por N_i, y del consumo energético de cada nodo, también en función de la carga soportada.

El comportamiento del tiempo de respuesta se mide previamente a la operación, mediante un experimento de carga sencillo con el servicio real que permita relacionar de forma directa el tiempo de respuesta esperado con la carga en número de sesiones activas. Esto permite obtener un modelo sencillo y global del sistema y con alta representatividad, pues proviene de datos empíricos. Este modelo medido a priori se considera el modelo inicial que se le proporciona al sistema para operar, y en la operación del algoritmo podría ser adaptado y actualizado dinámicamente si las características de la carga o de los nodos varían en tiempo de ejecución.

De esta forma, el tiempo de respuesta es medido para cargas de usuarios crecientes y ajustado mediante un modelo de estimación apropiado. El Gráfico 3 muestra tres ejemplos de modelos obtenidos para tres nodos de trabajo de características diferentes.

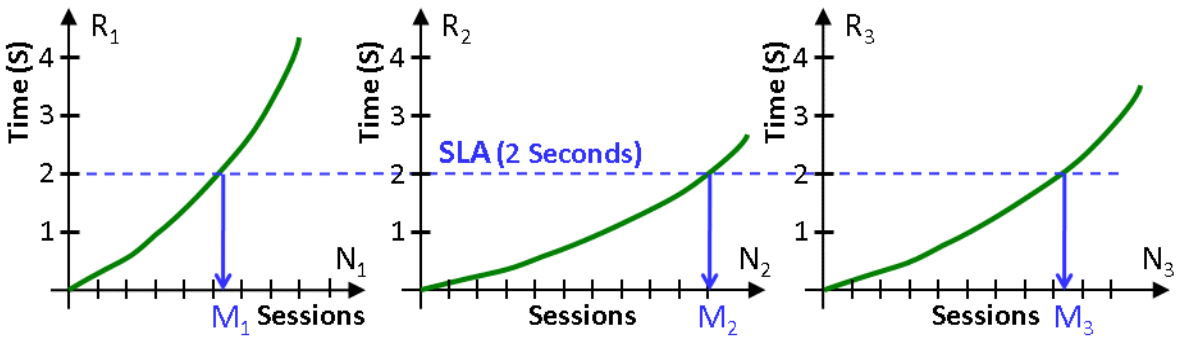

Gráfico 3: Ejemplos de estimación del tiempo de respuesta

Como se puede observar, en base al SLA (en la figura, 2 segundos como ejemplo) es posible calcular la capacidad que tiene un nodo sin llegar a violar el tiempo de respuesta límite (denotada por M_i). En el ejemplo, el nodo de mayor capacidad es el segundo. Para el cálculo de esta capacidad, teniendo la curva polinómica de ajuste, basta con ejecutar un procedimiento de cálculo de raíces analítico (será clave su optimización dada la importancia que tiene en la toma de decisiones la estimación de varias curvas).

Para el consumo energético de los nodos también se va a usar un modelo de estimación dinámico. Los computadores de hoy en día no consumen siempre la misma cantidad de energía y habrá que establecer una relación entre la carga que soporta un nodo y el consumo energético.

De los algoritmos estudiados, aquellos que tienen en cuenta de forma explícita el consumo energético, lo hacen mediante estimaciones de carácter teórico que, tras examinar y experimentar de forma empírica no se muestran como válidos para la optimización, porque no representan el consumo total del computador ni de los procesadores modernos con tecnologías de gestión avanzada. En el Gráfico 4 se muestran resultados para los tres ejemplos típicos de modelos de estimación de consumo.

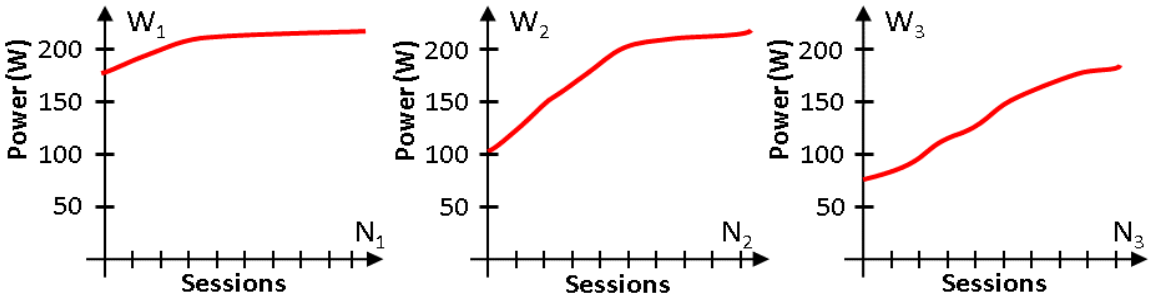

Gráfico 4: Ejemplos de estimación del consumo

Como se puede observar, en primer lugar, la forma de las curvas no tiene mucho que ver con los modelos cuadráticos teóricos que relacionan la frecuencia con el consumo mediante una relación casi directa. Estas curvas muestran el consumo básico (*idle power*) y el consumo de pico (*peak power*) y lo relacionan con una métrica directa y fácilmente obtenible por el módulo distribuidor, como es el número de sesiones concurrentes.

En el Gráfico 4, el primer nodo corresponde con un nodo poco eficiente (casi siempre se encuentra rozando el consumo de pico), mientras que los otros dos nodos son más eficientes.

El segundo nodo es el nodo paradigmático de un nodo con un muy buen control del *DVS* en el procesador, que permite escalar de forma lineal hasta que éste se satura o ya no tiene tiempo de reflexión entre las peticiones a atender y todas sus unidades funcionales están ocupadas, dando lugar a que el *DVS* no consigue obtener ahorros de ahí en adelante.

Finalmente, el tercer nodo es un claro ejemplo de un nodo muy eficiente, dónde además de una buena gestión de *DVS* en el procesador, hay necesariamente una buena gestión del resto de dispositivos, lo que permite crecer de forma lineal en consumo.

El consumo de pico, en general tiende a estabilizarse en un punto, de forma que incrementos de usuarios no suponen un incremento del consumo, lo que incrementa la eficiencia en las zonas de carga elevada.

De cara a analizar la eficiencia de cada nodo, además de buscar la curva con valores mínimos, es conveniente buscar dos aspectos: la mayor diferencia entre el consumo básico y de pico, lo que permitirá gestionar de forma más profunda el computador, y la mayor proporcionalidad, esto es, el incremento lo más lineal posible del consumo (o sublineal, si es posible, aunque todos los modelos tienden a ser supralineales). Esta proporcionalidad es la clave que permite optimizar el consumo.

A partir de estas curvas, medidas empíricamente, se pueden deducir otras curvas de eficiencia de forma fácil, y utilizar métricas de eficiencia, como se analizará posteriormente.

3.2 Estados de operación de los nodos

Con el objetivo de realizar la optimización del funcionamiento de los nodos, cada uno de ellos podrá pasar por un estado de funcionamiento diferente que determinará las operaciones que está permitido hacer con él y si acepta carga o no.

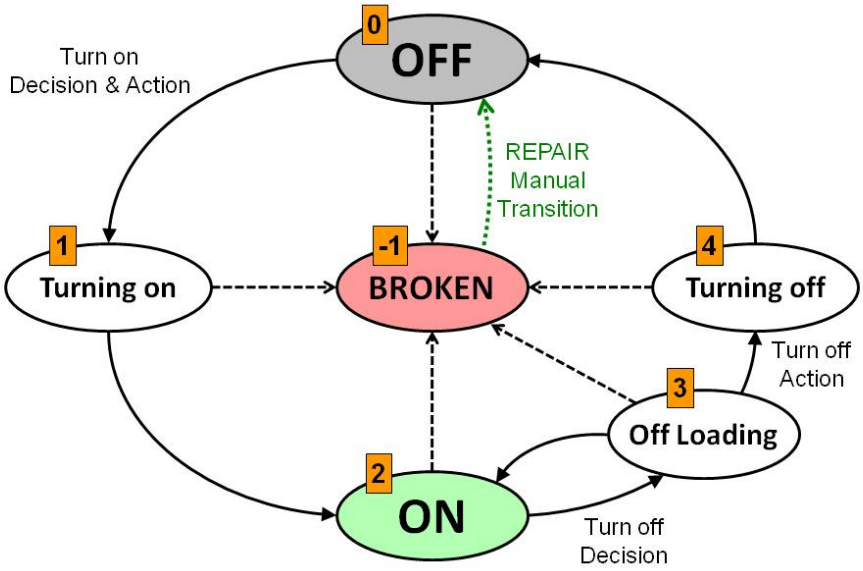

Gráfico 5: Diagrama de estados de operación de un nodo

El Gráfico 5 presenta el diagrama de estados de operación para un nodo del clúster (muestra adicionalmente el código numérico de cada uno de los estados) así como las transiciones permitidas.

Normalmente, los nodos estarán durante períodos largos en los estados estacionarios, marcados en color en el diagrama; estos estados, *ON* y *OFF*, además de *BROKEN*, son deseables que sean

estacionarios pues una minimización del número de transiciones minimiza la histéresis del algoritmo y conserva el hardware.

En la transición entre cada uno de los estados estacionarios, el nodo pasará por los diferentes estados transitorios y transiciones.

Cuando el gestor energético decide encender un nodo que está apagado (*OFF*), inmediatamente le envía un paquete mágico *WoL* y ejecuta la transición al estado transitorio *TURNING_ON*. Después de un periodo transitorio durante la recuperación o arranque del nodo, este notificará su disponibilidad al nodo distribuidor y pasará a estar operativo (*ON*) en estado estacionario. En este punto el nodo puede empezar a aceptar sesiones.

Si transcurrido un tiempo desde el envío del mensaje de encendido, el nodo no notifica su disponibilidad, el nodo será marcado como estropeado (*BROKEN*). En este estado el nodo está fuera de servicio y se deberá notificar al administrador del clúster el problema para que repare el nodo y éste vuelva al conjunto de nodos operativos.

Cuando el algoritmo decide apagar uno de los nodos, puesto que no se contempla la migración de carga, éste se pasa a un estado transitorio (*OFF_LOADING*) en el que no acepta sesiones y está planificado su apagado cuando se termine de ejecutar las transacciones que tiene asignadas. Cuando esto ocurre, pasa a un segundo estado transitorio (*TURNING_OFF*) tras enviarle una señal de apagado. Pasado un tiempo predefinido para el nodo, se marca como apagado (*OFF*).

Adicionalmente, cuando un nodo está en descarga y se decide encender un nodo, éste se puede marcar como encendido de nuevo, de forma que el rescate es inmediato y la respuesta del clúster cuando se producen incrementos de carga mejora. Además, se conserva el hardware reduciendo los apagados y encendidos.

Dado que siempre se escoge para apagar el nodo menos eficiente de los encendidos y para el encender el más eficiente de los apagados, se garantiza que el subconjunto de nodos encendidos siempre va a ser el más eficiente (más detalles en las secciones 3.3 y 3.4).

3.2.1 Bloqueos de transiciones de estados y protección de histéresis

Para minimizar las transiciones entre estados estacionarios y mejorar la respuesta, se han estudiado posibles bloqueos para ejecutar transiciones dependiendo del estado de otros nodos.

El primer bloqueo se produce en el encendido de un nodo, de forma que cuando hay un nodo en estado *TURNING_ON*, no se puede encender otro nodo. El motivo es sencillo y es que la capacidad del clúster aún no se ha modificado hasta que este nodo no se termine de encender y, por tanto, en nuevas aperturas de sesión se seguirá solicitando encender otro nodo, de forma que si el tiempo de encendido es alto, podrían ordenarse encender todos los nodos.

En cambio, en el caso del apagado, no ha lugar, pues la capacidad se modifica instantáneamente al poner un nodo en descarga, y dependiendo del tiempo de descarga, posiblemente desconocido, pueden empezarse a descargar otros nodos (sobre todo en transitorios de carga descendente muy rápidamente).

Sin embargo, existe una situación en la que tener varios nodos en descarga puede ser problemática, y es a la hora de sacar un nodo de descarga debido a un incremento repentino de la carga soportada por el clúster.

Extraer nodos del estado *OFF_LOADING* a *ON* es una medida que permite reducir mucho el tiempo de respuesta del algoritmo en casos de cambios bruscos de carga.

Para solucionar el problema de escoger qué nodo sacar de descarga cuando existen más de un nodo en dicho estado se pueden tomar dos opciones: sólo permitir la existencia de un nodo en ese estado, estableciendo un bloqueo similar al introducido en el encendido, lo que puede hacer que el algoritmo tarde en responder en decrementos rápidos de carga, haciendo que el clúster sea menos eficiente o, por otro lado, permitir varios nodos en descarga, pero de cara a seleccionar cuál hay que escoger para volver a poner en operación, aplicar el cálculo de consumos tentativos explicado en las secciones 3.3 y 3.4, pero reduciendo el conjunto de nodos candidatos al conjunto de nodos en descarga.

Para discriminar entre nodos de igual eficiencia que pueden ser *rescatados*, se prefieren los nodos que, teniendo el mismo consumo tentativo, tiene una mayor utilización, lo que minimiza el tiempo de balanceo de carga posterior al *rescate*.

Aplicar este cálculo permite garantizar que el conjunto de nodos encendidos siempre contiene a los nodos más eficientes, debido a que se apagan primero los menos eficientes y, en cualquier situación, se escogen para encender los más eficientes. Con las transiciones existentes, no es posible que se dé el caso en el que se encienda un nodo menos eficiente que otro que esté apagado, pues el segundo nodo ha de estar encendido previamente al primero.

3.3 Acciones cuando la carga aumenta

El Gráfico 6 resume las acciones que emprende el algoritmo cuando se produce un incremento de carga. En el momento en el que distribuidor recibe una petición de apertura de sesión, el algoritmo de optimización empieza a operar, decidiendo si se acepta la sesión y, en ese caso, a qué nodo se asigna. Además, puede ser necesario incrementar la capacidad del clúster debido a que ahora se está soportando una nueva sesión.

En primer lugar, cuando una sesión llega al distribuidor, se aplica el control de admisión, actualizando los datos de capacidad y utilización de cada nodo que se esté gestionando y esté encendido (aceptando sesiones), esto es, en estado *ON*.

Tal y como se observa en el Gráfico 5, existe la posibilidad de que, además de llegar nuevas sesiones que soliciten los clientes del servicio, lleguen sesiones que están en proceso de migración de otro nodo de trabajo.

Este el caso de las sesiones que están asignadas a un nodo que ha fallado y que están siendo reasignadas, por tanto, como ya se admitieron en su momento al servicio, no se aplica control de admisión en este caso (ver más detalles de la gestión de tolerancia a fallos en el capítulo 4).

Para el cálculo de la capacidad máxima se realiza el proceso analítico de cálculo de raíces para la curva estimada de respuesta y el SLA definido en el servicio, obteniendo M_i sesiones máximas para cada nodo i. Para calcular la utilización, teniendo en cuenta que se tienen abiertas N_i sesiones en el momento actual, la utilización del nodo i se define como:

$$U_i = \frac{N_i}{M_i}$$

De esta forma, se sabe qué proporción de sesiones hay actualmente en uso respecto al máximo de sesiones que permite soportar el SLA. Nótese cómo al utilizar un modelo sencillo pero empírico, no es necesario realizar uso de métricas de utilización de dispositivos ni latencias de red, pues todo viene representado en el modelo final y, además, es la visión del usuario final del servicio.

Para el control de admisión, si existe al menos un nodo con una utilización inferior a 1,0, se procederá admitir la sesión; en otro caso se rechaza la misma, pues todos los nodos están al máximo de capacidad que permite satisfacer el SLA.

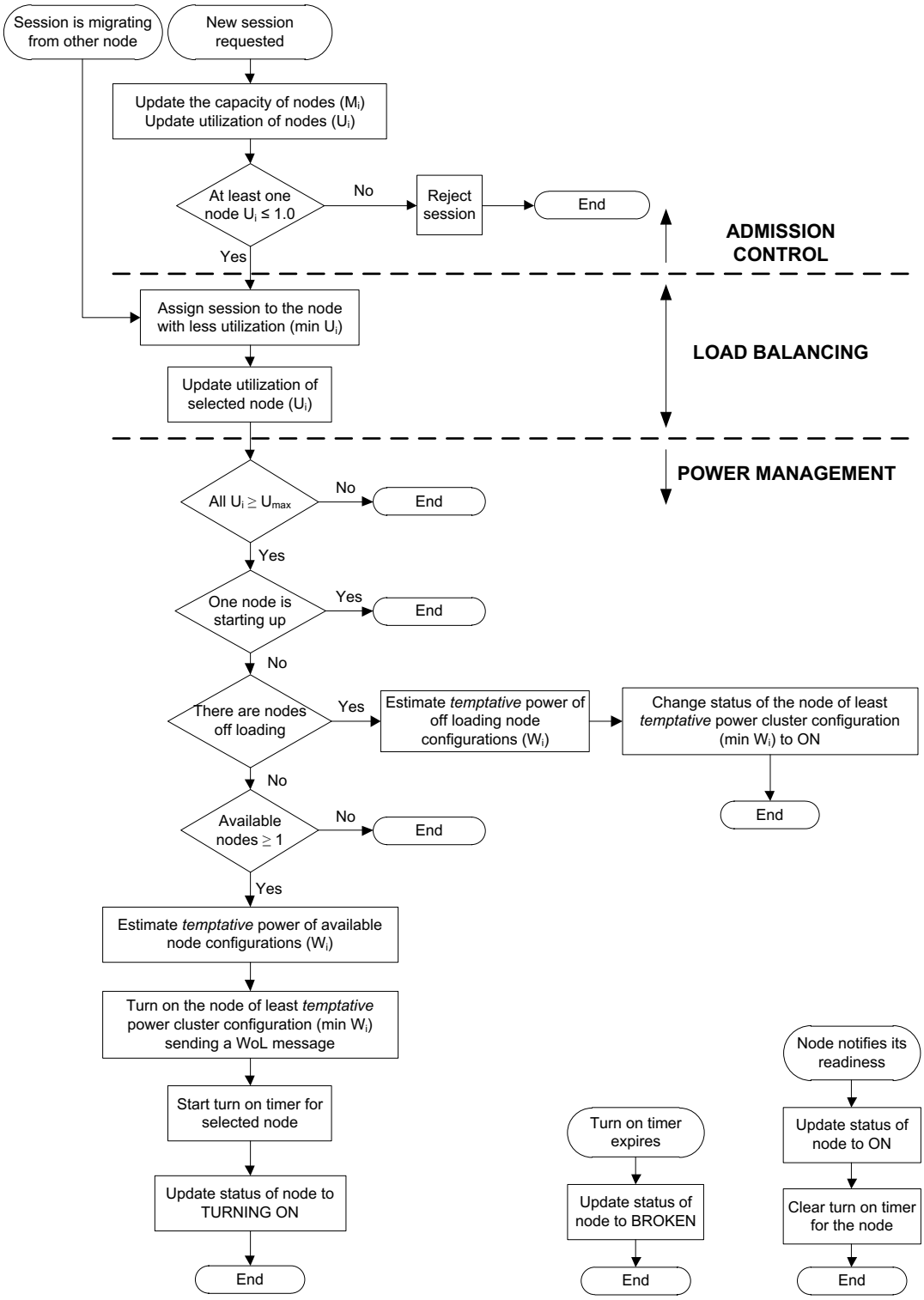

Gráfico 6: Diagrama de flujo para el incremento de carga

En caso de admisión, se asigna la sesión al nodo de menor utilización, de forma que se balancea la carga del sistema, se actualiza la utilización del nodo asignado y se inicia la fase de control de energía.

Estos primeros pasos permiten garantizar el cumplimiento del SLA, evitando una utilización excesiva de los nodos de computación. Los siguientes pasos permiten minimizar el número total de nodos encendidos y el consumo de los mismos, siempre manteniendo el SLA.

Para ello, el algoritmo comprueba que las utilizaciones de todos los nodos no sobrepasen un umbral definido como el parámetro del algoritmo U_{max}. Este parámetro debe ser inferior a 1,0, pero estar cercano, pues todo lo que se aleje de 1,0, será potencia de cálculo (y energía) desaprovechada. El escoger un valor u otro dependerá del tiempo de encendido de los equipos, que si se mantienen en suspensión en vez de apagados será menor (pero los nodos en estado *OFF* tendrán un consumo no despreciable).

En caso de que las utilizaciones superen el umbral U_{max}, el algoritmo solicitará al gestor de energía el encendido de un nodo, si es que hay alguno disponible en estado *OFF* u *OFF_LOADING*. Para seleccionarlo, se calculará el consumo de todas las configuraciones posibles con los nodos disponibles para el encendido, seleccionando el nodo que genere, para la carga existente, la configuración de mínimo consumo.

Esto garantiza tener siempre el conjunto de nodos más eficiente encendido de forma que el consumo es mínimo. Por último, se actualizan los estados y utilizaciones y se envía el mensaje *WoL*, estableciendo un temporizador de encendido y esperando por el encendido del nodo de forma asíncrona bloqueando, además, nuevos encendidos hasta que éste se encienda y adquiera, por tanto, el estado *ON*.

En este transitorio, dónde el encendido está bloqueado, pueden producirse rechazos de sesión temporales, por lo que el ajuste del umbral $U_{máx}$ es la clave para hacer al algoritmo reaccionar a tiempo y encender un nodo con antelación suficiente como para evitar rechazos indeseados, pero lo suficientemente tarde como para aprovechar la capacidad instalada en el clúster al máximo.

Cuando el nodo notifica su disponibilidad, se le comienzan a asignar sesiones y la capacidad del clúster es finalmente incrementada. Si se supera el tiempo límite de encendido, se asigna al nodo el estado *BROKEN*.

Si la notificación llega retrasada, el nodo vuelve a estado *ON*, pero se avisa igualmente al administrador para que compruebe el estado del hardware, que puede ser síntoma de un fallo inminente.

No obstante, las notificaciones retrasadas presentan un problema relativo al incremento de capacidad sin control del algoritmo: cuando el nodo es marcado como BROKEN, el algoritmo

escoge otro nodo para encender, si es que está disponible, de forma que incrementa la capacidad disponible. El problema se produce cuando, al llegar la notificación con retraso, el nodo que está marcado como *BROKEN*, pasa a estar en estado *ON*. En este momento se incrementa la capacidad disponible y el algoritmo, si la carga se mantiene, probablemente decidirá apagar otro nodo (o el recién encendido), de forma que se incrementa la histéresis del algoritmo.

Por esto es fundamental medir de forma correcta los tiempos de encendido y apagado de los nodos, además de incrementarlos en un margen de error apropiado.

El flujograma anterior cuenta con dos posible entradas, la destinada a la llegada de una nueva sesión, ya descrita, y la que corresponde a una sesión que se encuentra en migración desde uno de los nodos que ha fallado. En este caso, no se aplica control de admisión puesto que dicha sesión ya ha sido admitida anteriormente y sólo es necesario buscar un nuevo nodo donde ejecutarla. El algoritmo de reubicación de sesiones se encuentra descrito en el Capítulo 4.

3.4 Acciones cuando la carga disminuye

El momento en el que se cierra una sesión es el otro punto de entrada al gestor energético. En este caso, además de actualizar los estados de los nodos, se debe decidir si se reduce la capacidad del clúster después de la modificación sufrida en el estado global.

Las acciones a tomar son básicamente las opuestas a las tomadas para el encendido, pero con una ligera diferencia.

En el Gráfico 7 se muestra el diagrama de flujo de las acciones a tomar por el algoritmo en el caso de un cierre de sesión.

En este caso, se define el segundo y último parámetro de funcionamiento del algoritmo, el umbral de apagado U_{min}, que define una utilización tal, que si la utilización de todos los nodos está por debajo de ella, debe apagarse uno de los nodos. Esta utilización es una estimación de la utilización tras el balanceo de la carga que se produce al apagar un nodo, o utilización de *post-apagado*.

En la sección 3.5 se detalla el porqué de este comportamiento y las implicaciones que tiene, no obstante, como resumen de las mismas, el objetivo de este umbral dinámico es estabilizar la operación del algoritmo, concentrando la carga en los mismos niveles sea cual sea el número de nodos que componen el conjunto de nodos encendidos.

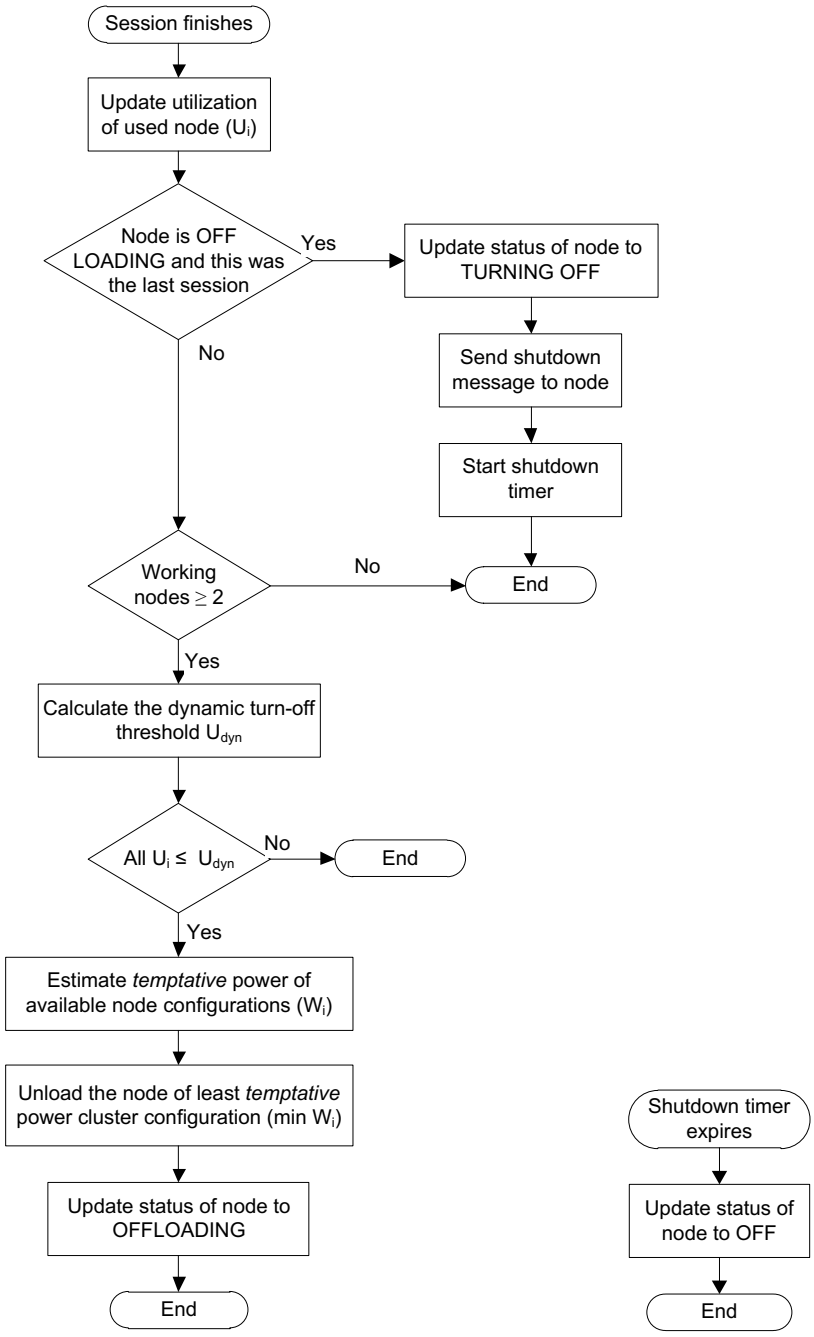

Gráfico 7: Diagrama de flujo para la disminución de carga

Si todos los nodos cumplen la condición de estar por debajo de U_{din} (el umbral de pre apagado dinámico), se decide apagar uno de ellos (quedándose siempre con uno disponible para la operación del clúster).

En este caso, se estima el consumo o eficiencia de los nodos en el momento actual (pueden estimarse tanto el consumo como la eficiencia, aunque no son términos exactamente equivalentes para este caso, ver sección 3.6) para todas las configuraciones posibles con los

nodos encendidos, seleccionando el nodo que hace que el consumo sea mínimo para la configuración del clúster con un nodo menos y la carga actual.

El motivo de esta estimación es que las curvas de carga esperadas de servicios transaccionales típicos son muy suaves y no cuentan con oscilaciones abruptas entre aperturas y cierres de sesión (en otras palabras, estas curvas cuentan con alta histéresis). De esta forma, cuando se producen incrementos de carga, es esperable que sigan produciendo incrementos y cuando se producen descensos, es esperable que se sigan produciendo descensos.

Una vez seleccionado un nodo, se actualiza su estado y se pone en descarga. Cuando se termina la descarga, se le envía un mensaje de apagado físico y se inicia un temporizador de apagado. Cuando este expire, se marca el nodo como apagado. No hay comprobación física de apagado pues no es necesaria para la operación del algoritmo. No obstante, si para un determinado hardware sobre el que opere el algoritmo es necesario, se cuenta con la posibilidad de sustituir este mecanismo por uno de *polling*.

3.5 Umbrales de funcionamiento

El algoritmo de optimización ve el clúster como una caja negra de cara a su modelado y las decisiones que se toman sobre esta caja negra son estrictamente dependientes de la garantía de la *QoS* del servicio. De esta forma, sólo se necesita configurar dos parámetros (además de proporcionar la descripción del hardware y del servicio que va a controlar mediante los modelos de *QoS* y potencia), que son los umbrales U_{max} y U_{min}. Estos dos parámetros definen cuándo se va a encender y apagar un nodo y, en definitiva, la precisión con la que el algoritmo va a funcionar.

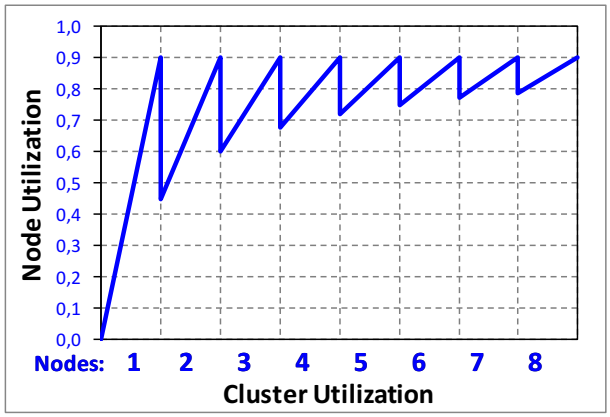

Gráfico 8: Utilización del clúster cuando la carga aumenta

En el Gráfico 8 se representa la evolución de la utilización combinada de un clúster teórico homogéneo utilizando un umbral U_{max} de 0,9. La representación permite comprender cómo evoluciona la utilización del clúster conforme se van enciendo nodos mientras la carga aumenta.

Hay que tener en cuenta que en la Gráfico 8 los balanceos y tiempos de encendido son instantáneos para enfatizar el comportamiento dependiente del umbral.

Cuando el primer nodo llega a superar el umbral de utilización máxima, U_{max}, se enciende el segundo, de forma que la utilización baja a 0,45 por nodo. Al encender el tercer nodo, la utilización baja a 0,33 por nodo, como era esperable.

Con esto, se puede calcular la utilización esperada del clúster después de encender un nuevo nodo y que se complete el balanceo de carga, si se opera con N nodos y se pasa a $N + 1$ nodos en un clúster homogéneo:

$$U_i = \frac{N}{N+1} \cdot U_{max}$$

Es remarcable cómo la selección de este umbral U_{max} ha de estar muy próxima a 1,0, pues, como se puede ver en el ejemplo, al utilizar 0,9, se desaprovecha sistemáticamente el 10% de la capacidad, que sólo se usará cuando ya no quedan nodos apagados y durante los transitorios de encendido de los nodos.

La relación entre el umbral de encendido y de apagado es también importante, pues el sistema ha de tener una respuesta que tienda a concentrar la carga y maximizar dentro de lo posible la utilización de los nodos de computación.

El umbral U_{min} representa la utilización que cada nodo del clúster debería de tener tras apagar un nodo para que se apague dicho nodo. Esto es, se realiza la comparación de cada una de las utilizaciones de los nodos tal y cómo deberían de quedar después de apagar el nodo candidato y realizar el balanceo de carga.

Como se puede observar, en el proceso de encendido se compara la utilización antes de encender con el umbral, mientras que en el proceso de apagado se compara con la utilización después de apagar. Esta diferencia es debida a que, si se quiere mantener la utilización por debajo de un umbral, es necesario saber cómo va a quedar después de apagar un nodo. Al encender no es necesario realizar el cálculo de la situación final, ya que tras encender se sabe que la utilización va a ser más baja. Al apagar, sin embargo, la utilización sube, con lo que puede superar el umbral objetivo.

Además, si se establecen umbrales iguales, se tiene el problema de la desaparición de la histéresis en el algoritmo, entrando en un ciclo estacionario de encendido y apagado infinito.

A partir del Gráfico 8 se puede comprobar cómo al establecer los dos umbrales iguales (de encendido y post-apagado) se entra en un círculo de encendido y apagado infinito, pues en cuanto un nodo alcanza la utilización de 0,9 (en el ejemplo) se encenderá un nuevo nodo al superarse el umbral de encendido. Pero además, tras el cierre de una sesión se apagará de nuevo, ya que la utilización pasa a estar por debajo del umbral de apagado establecido de forma estática. Utilizar umbrales diferentes y el uso de bloqueos de estados permite mitigar el problema, como se muestra en el Gráfico 9, en el que la línea verde muestra cómo se irían encendiendo nodos al aumentar la carga (yendo hacia la derecha en el eje X) y la línea roja cómo se irían apagando al descender la carga (yendo hacia la izquierda en el eje Y).

De esta forma, al superarse el umbral de encendido se incrementará la capacidad del clúster y, si la carga se reduce, no se reducirá la capacidad del clúster hasta que la utilización baje del umbral de apagado que, al ser menor al umbral de encendido, no provocará un ciclo de encendido/apagado constante.

Una diferencia grande en los umbrales reduce la probabilidad de reinicios, pero incrementa la capacidad desaprovechada del clúster.

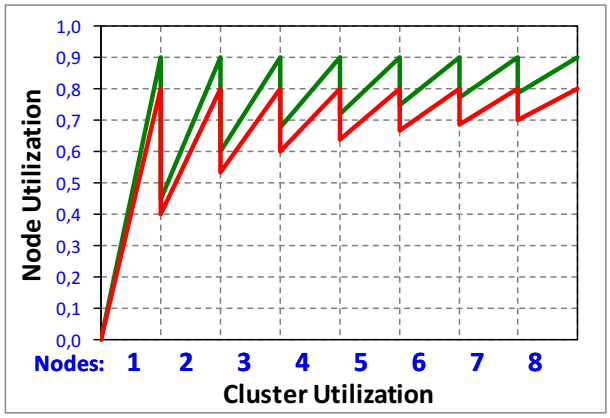

Gráfico 9: Comportamiento con umbrales diferentes

No obstante, aunque los umbrales sean diferentes, el umbral de utilización para el que hay que iniciar el apagado de un nodo no es siempre el mismo, y depende de la cantidad de nodos que estén encendidos. Tal y como puede verse en el Gráfico 9, la aportación a la capacidad global del clúster de cada nodo es proporcionalmente menor cuantos más nodos están encendidos. De esta forma, para eliminar el problema del ciclo de apagado sin fin, habría que establecer un umbral de apagado inferior a la utilización esperada para un número de nodos mínimo. En el caso de

una configuración de 2 nodos con $U_{max} = 0{,}9$ equivaldría a establecer un umbral de apagado a 0,45. Esto eliminaría el problema del ciclo de apagados, pero volvería al algoritmo extremadamente ineficiente en situaciones de descenso de carga.

Si se sigue el Gráfico 9 de derecha a izquierda, puede observarse cómo, en una situación de carga descendente, el utilizar un umbral de apagado estrictamente menor al umbral de encendido es fundamental para resolver el problema de los ciclos de encendido/apagado. Por ejemplo, cuando se está trabajando con 2 nodos, si se utilizase como umbral de encendido y apagado el mismo valor, 0,9, en cuanto descendiese la carga por debajo de ese valor, por ejemplo a 0,89, se apagaría un nodo, pero la carga del nodo restante estaría automáticamente por encima de 0,9, ya que tiene que asumir la carga del otro nodo, con lo que inmediatamente se iniciaría el encendido del nodo apagado.

La clave para resolver el problema de forma eficaz es darse cuenta de que el valor del nivel de carga que permite obtener un umbral después de apagar no es constante, sino que depende del número de nodos encendidos (y de sus capacidades en clúster heterogéneos) en cada instante. Siguiendo la línea roja de derecha a izquierda en el Gráfico 9 se puede observar cómo el umbral para apagar un nodo y que la utilización quede en un umbral inferior al umbral de apagado $U_{mín}$ (en este caso, 0,8), es distinto según el número de nodos encendidos. Por ejemplo, para pasar de 8 nodos a 7, la utilización antes de apagar un nodo tiene que bajar por debajo de 0,8. En cambio, para pasar de 3 nodos a 2, la utilización antes de apagar tiene que bajar por debajo de 0,55. En los dos casos, después de apagar se consigue estar en el umbral objetivo de 0,8. Por lo tanto, se tienen dos umbrales de apagado: el de pre-apagado, que es dinámico, y el de post-apagado, que es estático y es el umbral $U_{mín}$ que se fija como parámetro del algoritmo.

Cuando se cierra una sesión se calcula, en base al umbral de post-apagado $U_{mín}$ configurado en el algoritmo, el umbral de pre-apagado dinámico U_{din}, que depende del número de nodos encendidos, de forma que para un clúster homogéneo:

$$U_{din} = \frac{N-1}{N} \cdot U_{min}$$

Una vez calculado el umbral de pre-apagado dinámico, se conoce cuál es la utilización por debajo de la cual deben estar todos los nodos para poder apagar uno. A continuación, se escoge qué nodo apagar mediante la estimación de consumo.

3.6 Consideraciones para clústeres heterogéneos

En clústeres heterogéneos, donde cada nodo tiene distinta capacidad y distinto consumo para una misma capacidad, es necesario modificar dos aspectos de la estrategia presentada anteriormente: el algoritmo de selección de nodo y el cálculo de los umbrales. En esta sección se estudian estos dos aspectos.

3.6.1 Algoritmo de selección de nodo

Si se está manejando un clúster homogéneo, donde todos los nodos tienen la misma capacidad y consumo, el algoritmo para seleccionar qué nodo encender o apagar funciona como un algoritmo de "seleccionar el primero", pues todas las opciones son iguales.

En el caso de clústeres heterogéneos, el manejo de las curvas de potencia y de respuesta independientes para cada uno de los nodos del clúster es fundamental para optimizar el funcionamiento; por tanto, se ha extendido el algoritmo para el manejo de la diferentes curvas a nivel de nodo haciéndolo compatible con la administración de clústeres heterogéneos, donde cada nodo tiene parámetros diferentes al resto de nodos.

Opcionalmente, puede decidirse qué parámetro energético optimizar, o bien directamente el consumo, o la eficiencia. El consumo mide el gasto energético de un nodo sin tener en cuenta la capacidad del mismo, mientras que la eficiencia permite conocer el consumo energético para cada sesión que soporta dicho nodo, de esta forma se puede optimizar la topología del clúster de forma más detallada (por ejemplo, los nodos de baja capacidad y alto consumo tendrán menos eficiencia y se usarán menos que nodos con bajo consumo o nodos de muy alta capacidad, que permite tener mayor eficiencia). El utilizar el consumo directamente hace que se usen topologías de consumo mínimo, pero no permite garantizar que éstas tengan una eficiencia alta, o lo que es lo mismo, que para un consumo energético determinado, se tenga un nivel alto de capacidad, lo que efectivamente, puede llevar a topologías que no tengan en consumo mínimo, ya que pueden haberse encendido nodos de consumo mínimo individualmente, pero más cantidad de ellos de los que se hubieran encendido con una optimización de eficiencia.

Los objetivos de optimización de eficiencia pueden ser variados: además de minimizar el consumo por sesión, podría fijarse el consumo máximo y tener que maximizar la capacidad para dicho consumo, o fijar una capacidad mínima del servicio y tener que minimizar el consumo.

Fundamentalmente, se puede establecer una métrica de eficiencia en términos de energía por sesión (con objetivo de su minimización), de forma que, aunque la tendencia va a ser la concentración de carga igualmente, el conjunto de nodos seleccionando puede variar con respecto al seleccionado utilizando una métrica de consumo absoluto para la misma carga o un objetivo de optimización diferente.

Si se usa una métrica de eficiencia energética, típicamente se escogerán los nodos de mayor capacidad[1], con el fin de minimizar el impacto del consumo de base de cada nodo por sesión, mientras que con una métrica de consumo absoluto, siempre se minimiza el consumo total, lo que dependerá del modelo de potencia obtenido para cada nodo.

Es importante indicar que para las mediciones de eficiencia es conveniente utilizar métricas que relacionen la energía consumida con el número de sesiones Si se usan medidas de potencia, se están utilizando métricas de consumo instantáneo, que puede tener diferentes características según el servicio proporcionado.

3.6.2 Cálculo de umbrales

En el desarrollo del proyecto fin de carrera donde se diseñó y validó el algoritmo de optimización energética [Medrano11] se utilizó una aproximación de $U_{mín}$ calculada de forma aproximada. En esta aproximación, para el cálculo de $U_{dín}$ se utilizaba el número de nodos en lugar de las capacidades y utilizaciones de los mismos, de forma que no se tenía en cuenta de forma correcta las diferencias de capacidad de cada nodo.

En un clúster heterogéneo, el umbral $U_{dín}$ es distinto para cada nodo, en concreto, cada vez que se necesita hallar $U_{dín}$ porque se ha cerrado una sesión y se va a tomar la decisión se apagado hay un valor de $U_{dín}$ para cada tipo de nodo que se tenga, dado que la capacidad de cada uno es diferente y el cambio que va a producir su apagado en el clúster es diferente. Utilizar el número de nodos de forma directa es una aproximación equivalente en el caso de clústeres homogéneos, pero no lo es en el caso de un clúster heterogéneo.

[1] Aunque lo habitual es que los nodos con mayor capacidad tengan más eficiencia, no tiene por qué ser así, especialmente para nodos con capacidades no muy distintas, en los que un nodo más moderno, pero con menos núcleos de procesamiento, es posible que consiga con la tecnología mejorar la eficiencia con inferior capacidad. Fundamentalmente, el responsable de que los nodos de mayor capacidad sean más eficientes es el consumo *idle*, muy similar entre todos los tipos de nodos.

Esto presenta un funcionamiento correcto en clústeres homogéneos, ya que al ser todos los nodos iguales, el gestor va a operar de forma equivalente a un algoritmo round-robin, apagando y encendiendo los nodos por orden, puesto que las eficiencias y umbrales de pre-apagado son iguales para todos. Sin embargo, para clústeres heterogéneos no se realiza una operación correcta, ya que se tenderá a apagar nodos de baja capacidad sin tener en cuenta su eficiencia energética en el nivel de utilización adecuado, pues se utiliza un nivel de utilización de pre-apagado incorrecto.

Además, en [Medrano11] se calculaba $U_{dín}$ una sola vez para todos los nodos, no teniendo en cuenta las diferencias que cada tipo de nodo presenta en términos de capacidad, lo que es válido para clústeres homogéneos, pero no para clústeres heterogéneos.

En este trabajo se ha implementado el manejo de $U_{mín}$ de forma completa, esto es, calculada de forma dinámica para cada tipo de nodo y situación de carga utilizando las cuentas de sesiones activas completas y no las fracciones de utilización, además, se calcula el umbral para cada nodo y no una vez para cada uno de los nodos, lo que hace que el uso del gestor energético cuente con toda la precisión para cada uno de los nodos en el manejo energético sea cual sea la composición del clúster. En concreto, $U_{dín}$ se calcula de la siguiente forma:

$$U_{dín}^{(i)} = \frac{S - S^{(i)}}{S} \cdot U_{min}$$

Donde $U_{dín}^{(i)}$ es el umbral de pre-apagado dinámico en el caso de apagar el nodo i, S la cantidad de sesiones activas en total en el servicio y $S^{(i)}$ la cantidad de sesiones activas para el nodo i.

Como se puede observar, este umbral dinámico depende del nodo a apagar, concretamente de la cantidad de sesiones que soporte en el momento del cálculo y de la cantidad de sesiones que haya en el servicio, de esta forma, se puede calcular de igual forma para clústeres heterogéneos y homogéneos y permite predecir la utilización que debe de tener el nodo para que al apagarlo se alcance el objetivo de $U_{mín}$.

En la sección 5.2 se detallan los casos de prueba para estas configuraciones heterogéneas en condiciones similares a los experimentos diseñados en el apartado de pruebas y validación del proyecto fin de carrera para comprobar su correcto funcionamiento.

4 ESTRATEGIA DE TOLERANCIA A FALLOS

Con el objetivo de implementar un mecanismo de auto-optimización autonómico, además de la optimización del consumo energético es necesario proporcionar mecanismos de tolerancia a fallos que permitan la autogestión del clúster desde el punto de vista de la recuperación ante fallos en los nodos trabajadores. En este apartado se desarrolla el nuevo gestor de tolerancia a fallos integrado en el gestor autonómico, con el objetivo de maximizar la disponibilidad del servicio manteniendo las restricciones de calidad de servicio y minimización del consumo.

4.1 Fallos en sistemas distribuidos

Los sistemas distribuidos tienden a ser sistemas complejos, con muchas dependencias entre diversos agentes, elementos pasivos y actuadores.

Aunque que el sistema completo esté disponible, esto no implica que la calidad del servicio y la capacidad total se mantenga. En un clúster de balanceo de carga, si uno de los nodos deja de estar disponible, o bien para la misma cantidad de usuarios se cuenta con menores prestaciones, o bien la capacidad del servicio ha de reducirse con el fin de soportar a los clientes con una calidad de servicio mínima.

Otros problemas relativos a la pérdida de capacidad pueden relacionarse con la pérdida de capacidades funcionales del sistema, tales como la pérdida de un nodo especializado en el tratamiento de clientes de alta prioridad, que si bien pueden ser atendidos por los demás nodos del clúster, la calidad del servicio puede verse perjudicada.

4.1.1 Detección y recuperación de fallos

La detección de fallos en sistemas distribuidos es fundamental de cara a la implementación de mecanismos de tolerancia a fallos de forma autonómica.

La detección de fallos puede ser de dos tipos:

a) Detección prematura, de forma que se descubre la posibilidad de la existencia de un fallo antes de que se manifieste.
b) De recuperación, donde ya se ha producido el fallo y sólo que la posibilidad de paliar los efectos.

En general, la forma más extendida de detección es la segunda, de manera que una vez se produce el fallo de forma manifiesta, éste se detecta. Implementar políticas de detección prematura requiere de técnicas heurísticas que no siempre son totalmente precisas.

La técnica para la detección de fallos depende en gran medida de las características del sistema distribuido: se pueden detectar errores de forma síncrona, de forma asíncrona, mediante excepciones o interrupciones e incluso mediante eventos de temporización.

Dependiendo de las técnicas particulares a implementar y del sistema distribuido, la recuperación de fallos podrá ser llevada a cabo usando una técnica u otra (ver sección 4.1.3).

En el plano más puramente tecnológico, la detección de fallos puede realizarse usando diferentes patrones y metodologías de diseño de sistemas.

Por ejemplo, una forma muy extendida de implementación de sistemas de detección de fallos en sistemas distribuidos es mediante un control de latido (*heartbeat*). Desplegar este mecanismo requiere que todos los nodos trabajadores, en el caso de un clúster de balanceo de carga cuenten con un agente que envía notificaciones de disponibilidad con una frecuencia constante, de forma que cuando el nodo central que las recibe (por ejemplo, el nodo distribuidor de carga) determina que un nodo ha fallado tras no recibirlas durante un intervalo de tiempo.

Este tipo de mecanismos activos permite detectar los fallos con cierta antelación, que depende de la resolución temporal que se utilice para el enviar el latido, sin necesidad de esperar a acceder a los recursos para la detección del fallo.

Sin embargo, requieren de la dedicación de recursos del clúster para el control del correcto funcionamiento de los nodos, además de que, en situaciones de alta concentración de carga, la frecuencia de acceso a los nodos es superior, generalmente, al propio latido, por lo que las

detecciones se realizan en una mayor proporción al tratar de acceder a los recursos no disponibles.

Otras tecnologías de interés para la detección de errores remotos son las basadas en entrada/salida asíncrona, por ejemplo para el uso de *sockets* TCP/IP. Existen librerías que permiten realizar operaciones asíncronas sobre conexiones de red, de forma que no sólo se pueden paralelizar las operaciones de entrada/salida con computación, sino que además se pueden recibir notificaciones fuera de banda relativas a fallos y roturas de conexiones.

El gestor de tolerancia a fallos implementado se basa en un mecanismo de recuperación de fallos, debido a que requiere de menos recursos para la detección de errores y, realizando más de una petición por segundo, como se espera en un sistema de alta utilización, la frecuencia de peticiones es mayor que la que un posible mecanismo de latido pueda ofrecer.

4.1.2 Punto único de fallo

El punto único de fallo se define como aquel elemento o elementos del sistema cuyo fallo implica el fallo de todo el sistema. Los elementos que constituyen el punto único de fallo han de estar situados en serie con respecto al subsistema de más alto nivel cuando se usa un diagrama de bloques para modelar la ocurrencia de fallos y establecen una cota superior a la fiabilidad total del sistema.

Por ejemplo, en un clúster de balanceo de carga, un punto único de fallo común es el nodo que se encarga de realizar el balanceo o reparto de carga. Una estrategia para incrementar su fiabilidad es convertir este nodo en un subclúster, pero orientado a la fiabilidad, donde todos los elementos estén replicados. Otra estrategia que no involucra la réplica de hardware es el rediseño del sistema distribuido para convertirlo en un sistema no centralizado o *peer to peer* si es posible dadas las restricciones del problema.

Como conclusión, es necesario remarcar cómo es necesario evitar los puntos únicos de fallo de cara a proporcionar mecanismos de tolerancia a fallos, pero esto no se consigue de forma exclusiva mediante réplicas de elementos de hardware, sino también con diseños y mecanismos de software diseñados para la tolerancia a fallos.

El gestor autonómico descrito tiene como punto único de fallo el repartidor de carga, que deberá situarse en un clúster de tolerancia a fallos o replicarse mediante mecanismos de round-robin DNS, por ejemplo.

4.1.3 Recuperación de fallos, garantía de SLA y gestión energética

Tal y como se menciona en la sección 2.3, existen múltiples mecanismos de recuperación de fallos en clústeres de balanceo de carga. La tolerancia a fallos es un pilar importante dentro de la calidad de servicio apreciada por los clientes, que, además del tiempo de respuesta incluye el hecho de que el servicio esté disponible y oculte sus fallos a los clientes.

En sistemas transaccionales, los modelos de recuperación de fallos más comunes se basan en la migración de sesiones a otros nodos trabajadores. Adicionalmente, para recuperar una sesión de forma más rápida, se pueden combinar con mecanismos de *checkpointing* (por ejemplo, como los *redo logs* de los sistemas de bases de datos) y de sobre provisión (como los sistemas de réplicas de MapReduce), con la contra partida de multiplicar el coste en términos de recursos necesarios para atender cada sesión.

El principal problema a resolver por un sistema de gestión autonómica que garantice el cumplimiento del SLA a los clientes es, precisamente, seguir garantizando el SLA en momentos en los que uno o varios de los nodos han fallado y la capacidad del sistema se ve afectada.

En la práctica, en estos momentos el cumplimiento del SLA no se puede garantizar de forma plena en todos los casos sin añadir redundancia (incrementando los costes de operación y el consumo energético). Si bien es cierto que cuando la utilización del clúster no es completa, la pérdida de uno de los nodos trabajadores es suplida de forma inmediata por el resto, en situaciones de utilización plena la capacidad se reduce por debajo de la utilización y o bien se reduce la calidad de servicio o bien se abortan sesiones ya abiertas.

En cualquier caso, la calidad de servicio percibida por los clientes se resiente de alguna forma. Yendo más allá, en un sistema con gestión autonómica de energía, donde puede haber nodos sin ser utilizados, éstos van a estar apagados, lo que en caso de fallo de uno de los nodos activos introducirá retardos en la operativa del clúster.

4.2 Estrategia de tolerancia a fallos autonómica

El objetivo de un sistema autónomo tolerante a fallos es mantener el funcionamiento del sistema en la mayor cantidad de situaciones posibles, lo que incluye el fallo de uno o más elementos y, además, mantener la calidad de servicio y el máximo de capacidad posible mientras el problema se resuelve.

Adicionalmente, la operación autonómica debe de ir dirigida por la optimización energética y la garantía de la calidad de servicio.

En esta sección se describe la estrategia de tolerancia a fallos integrada con el sistema de gestión energético autonómico de forma que se trata de garantizar la disponibilidad del servicio manteniendo la calidad del servicio y minimizando el consumo energético.

4.2.1 Primera fase: detección del fallo

El primer paso en una estrategia de tolerancia a fallos es la detección de los mismos. Existen múltiples métodos para la detección de fallos, que dependen del modelo de software y topología de clúster que se esté manejando:

4.2.1.1 Detección asíncrona

Este método es más común en servicios concurrentes, de forma que los problemas se detectan de forma distribuida y no necesariamente en el momento en el que se producen. Coordinar los cambios que la estrategia de gestión autonómica debe aplicar en el sistema en estos casos es más complejo, de forma que, por ejemplo, en sistemas multiproceso, donde la concurrencia es un factor de complejidad, la aplicación de medidas, como la migración de sesiones a otro nodo de trabajo, puede llevarse a cabo parando todo el servicio o de forma progresiva en cada proceso.

En la estrategia desarrollada en el trabajo, la detección de los fallos es de carácter síncrono para cada cliente, de forma que la propia capa de red encapsula la detección de fallos mediante la rotura de conexiones TCP.

La detección del primer fallo es síncrona y en tiempo, esto es, en el mismo momento en el que uno de los hilos de servicio del repartidor de carga detecta que un nodo trabajador ha fallado desencadena un proceso de reajuste de la topología del clúster para hacer frente al fallo.

Sin embargo, la aplicación de las medidas, principalmente la reconfiguración de la topología del clúster y la reasignación de sesiones a otro de los nodos trabajadores, se realiza de forma asíncrona, en función de cada uno de los hilos de procesamiento de sesiones.

De esta forma, el re-balanceo de la carga es más progresivo y adaptativo a la carga en el momento del fallo. Adicionalmente, permite tomar ventaja de los nuevos nodos que potencialmente se añadan al clúster al no mover toda la carga en un mismo momento.

4.2.1.2 Detección síncrona

Se trata del método más sencillo. Está basado en servicios bloqueantes (síncronos) de forma que los problemas se detectan en el mismo momento en el que se producen o con una latencia mínima. Además, es posible aplicar medidas correctoras en ese momento, resolviendo el problema de forma completa.

El gestor de tolerancia se basa en una detección síncrona, debido a la sencillez de implementación usando entrada/salida síncrona y a la eficacia que ofrece para la detección de fallos conforme existe o no demanda por parte de los clientes del servicio, esto es, los fallos se detectan al solicitar una interacción uno de los clientes.

4.2.2 Segunda fase: migración y balanceo de sesiones

El balanceo de carga es una componente fundamental de un sistema transaccional que se ejecuta sobre clústeres. En el momento en el que se detecta un fallo, la primera sesión marcará dicho nodo como *BROKEN*. De esta forma no existe posibilidad de utilizar este nodo más y los administradores del servicio recibirán la notificación para reparar el nodo.

En este momento comienza un proceso distribuido asíncrono para rebalancear la carga del clúster, esto es, mover las sesiones en ejecución asignadas al nodo que acaba de estropearse a otros nodos de forma transparente para los usuarios finales.

Cada hilo de procesamiento de sesiones solicitará una reubicación de su sesión, de forma que se realiza un proceso similar al del balanceo de carga para obtener un nodo, seleccionando el de mejor eficiencia energética, donde ejecutar la sesión, con la salvedad de que el rechazo de sesión está deshabilitado.

No es posible rechazar una sesión, pues ya fue aceptada en su momento. Sin embargo, como se puede deducir, si se tiene menos capacidad y el número de sesiones no va a decrecer (salvo por las sesiones que terminen), es posible violar la restricción del SLA en estos momentos.

Este proceso distribuido permite rebalancear la carga de forma adaptativa sin necesidad de bloquear todo el servicio para reconfigurarla de forma centralizada, mejorando la disponibilidad de los nodos que están funcionando correctamente.

Aunque el tiempo de adaptación puede ser mayor con esta estrategia, la reasignación es siempre bajo la demanda marcada por la actividad de la sesión de forma que sólo se toman acciones cuando es necesario.

Finalmente, existen dos aproximaciones para completar la migración de la carga asignada a uno de los nodos que ha fallado: ésta se puede migrar de forma completa o progresiva.

Una migración completa implica trasladar todas las sesiones que están asignadas al nodo que ha fallado de una sola vez a otro de los nodos operativos. Este proceso podría requerir del bloqueo temporal de la aceptación de sesiones y es un proceso corto.

Por el contrario, una migración progresiva se basa en mover las sesiones una por una a otro de los nodos de forma asíncrona. Este proceso es más largo, pero sin embargo, optimiza el balanceo de la carga y, más importante, no genera picos repentinos en la utilización de los nodos del clúster, que son problemáticos en términos de variación del tiempo de respuesta.

En el algoritmo de reubicación de carga se ha optado por esta segunda aproximación, debido a las importantes ventajas que aporta en términos de equidad en el balanceo de la carga extraordinaria que reciben los nodos activos cuando uno falla. Este balanceo es clave para, mientras se concentra la carga, todos los nodos se mantienen en niveles de utilización que permitan satisfacer los requisitos del SLA o, durante el tiempo transitorio de adaptación, tener las mínimas violaciones del mismo.

4.2.3 Tercera fase: reconfiguración de la topología

Debido a que se puede violar la restricción de garantía del SLA, junto con la reasignación de carga es necesario reconfigurar la topología del clúster para minimizar el tiempo en el que el servicio no está satisfaciendo los requisitos del servicio.

Fundamentalmente, cuando uno de los nodos se cae, es necesario incrementar la capacidad del clúster para proveer el servicio a los clientes de forma que sea posible seguir satisfaciendo las restricciones de calidad de servicio.

Dependiendo de la situación del clúster, será necesario encender o no nuevos nodos. En situaciones en las que los parámetros de encendido sean altos y los de apagado sean bajos, es posible que pese, a la pérdida de uno de los nodos, el clúster siga en los parámetros de calidad de servicio adecuados.

Matemáticamente, se puede definir por capacidad a reasignar, CR, a la cantidad de sesiones que están asignadas al nodo que ha fallado. Éstas serán las sesiones que necesitarán moverse a otro nodo trabajador y a las que progresivamente se irá aplicando la técnica de reasignación para ir cambiando una a una de forma balanceada.

En concreto, sea $S^{(i)}$ el número de sesiones que se están ejecutando en el nodo i y sea $C^{(i)}$ la capacidad de dicho nodo i, esto es el número máximo de sesiones que puede soportar sin violar el SLA de acuerdo a los modelos que describen su tiempo de respuesta en función de la carga.

Se puede definir por CD_E la capacidad disponible en estado E, siendo E un estado donde se tiene carga en el nodo (*ON* y *OFFLOADING*).

De esta forma, se puede calcular CA_S como la suma, para todos los nodos i en estado S, de capacidad máxima que permite satisfacer el SLA del nodo i-ésimo en ese estado menos el número actual de sesiones en dicho nodo:

$$CA_S = \sum_{i \in S} \left(C^{(i)} - U^{(i)} \right)$$

En principio, los estados interesantes para la gestión del fallo son *ON* y *OFFLOADING* ya que son los estados donde existe carga en los nodos. Nótese cómo para cualquier otro estado S′ distinto a *ON* y *OFFLOADING*, se define que $CA_S, \ = \ 0$, puesto que el nodo no es utilizable.

En este sentido, y en base a estos parámetros de capacidad, es posible aplicar un algoritmo que calcule cuántos nodos, y cuáles, es necesario encender.

Siendo CA_{ON} la capacidad en estado *ON* definida de la forma anterior y $CA_{OFFLOADING}$ la capacidad en estado *OFFLOADING*:

```
si CR ≤ CA_ON:
      No es necesaria ninguna acción (1)
si no:
      Es necesario activar más nodos (2)
```

Si la capacidad disponible en estado *ON* es suficiente para dar cabida a todas las sesiones que van a solicitar una reasignación (el algoritmo escogerá la rama `(1)` del pseudocódigo anterior),

no es necesario aplicar ninguna modificación a la topología del clúster, pues progresivamente se irán reasignando en las máquinas existentes.

El proceso de reconfiguración de la topología del clúster se ejecuta en cada momento en que una sesión es reubicada a un nuevo nodo. Como es natural, en esta situación, pueden llegar nuevos clientes al clúster y hacer que la capacidad en estado *ON* no sea suficiente para dar cabida a las nuevas sesiones y a la capacidad necesaria para las sesiones a reubicar.

Para evitar este problema existen dos mecanismos aplicados en este algoritmo:

1. El algoritmo de reconfiguración de la topología se aplica siempre que una sesión solicita ser reubicada. De esta forma, el algoritmo es totalmente adaptativo a los cambios de carga en momentos de fallo de nodos.

El control de admisión está desactivado para sesiones ya aceptadas, pero no para nuevos clientes. Con este mecanismo se minimiza las violaciones de SLA sólo para las sesiones ya aceptadas. Si se aceptaran nuevas sesiones adicionales, los ratios de violación del SLA se multiplicarían.

En el caso de que sean necesarios más nodos debido a que la capacidad en estado *ON* no es suficiente, será necesario activar más capacidad. Esta capacidad puede venir de nodos apagados o, si están disponibles, de nodos en descarga (*OFFLOADING*).

La reactivación de los nodos en descarga es inmediata y permite recuperar las condiciones de calidad de servicio de forma transparente para los nuevos clientes y con el mínimo de tiempo de espera para los clientes existentes (sólo es necesaria la reubicación de la sesión).

Además, puesto que el algoritmo garantiza siempre que el conjunto de nodos es el más eficiente, escoger los nodos en descarga permite incrementar la capacidad del clúster con nodos más eficientes.

La capacidad que es necesario activar, CE, se puede definir cómo la capacidad que no ha podido ser absorbida por los nodos encendidos:

$$CE = CR - CA_{ON}$$

Esto es, sólo será necesario activar nodos para la capacidad requerida por las sesiones a reasignar que los nodos ya encendidos no puedan acoger:

```
si CA_OFFLOADING > 0:
    Activar nodos para mín(CE, CA_OFFLOADING) sesiones (3)
CO := CE − CA_OFFLOADING
```

```
si CO > 0:
      Encender nodos para CO sesiones (4)
```

Si hay nodos en descarga, y éstos pueden hacerse cargo de las sesiones que no son soportadas por los nodos encendidos, no será necesario encender ningún nodo más, sólo activar nodos que estén en descarga para atender CE sesiones más de las que ya tienen actualmente.

Si no hay suficientes nodos en descarga para soportar toda la capacidad a activar, será necesario encender nodos que están apagados para proveer el servicio a las sesiones que ni los nodos encendidos ni los nodos en descarga en su totalidad pueden atender, esto es, CO sesiones:

$$CO = CE - CA_{OFFLOADING} = CR - CA_{ON} - CA_{OFFLOADING}$$

En este caso, debido a que los nodos necesitan un tiempo para estar disponibles desde la orden de encendido es necesario aplicar la técnica de bloqueo de encendido de nuevos nodos en el gestor energético.

El caso más desfavorable es que no sea suficiente con los nodos que están apagados, porque no haya bastantes. En este caso, la capacidad total de clúster se ha reducido y no se podrán atender a todos los clientes.

Las sesiones ya aceptadas tendrán tiempos de respuesta superiores al SLA y ninguna nueva sesión se podrá admitir mientras esta situación continúe. Cuando las sesiones vayan completándose y se normalice la capacidad atendida se volverá a satisfacer el SLA.

4.2.4 Decisiones de encendido en el gestor de tolerancia a fallos

Tal y como se observó en la sección 4.2.3, en el peor de los casos será necesario encender nuevos nodos, lo cual lleva un tiempo de espera del orden de minutos.

En este tiempo es posible que nuevos clientes soliciten acceder al clúster o que sesiones asignadas a nodos fallidos soliciten una reubicación. Ambas acciones desencadenan el proceso de decisión en el gestor energético que puede terminar ordenando el encendido de un nuevo nodo.

El proceso de reconfiguración de la topología tiene la potestad de encender nodos de forma extraordinaria; mientras esto ocurre, es necesario bloquear decisiones de encendido para evitar encender todos los nodos disponibles mientras su capacidad no está disponible, bien por una nueva sesión entrante o por una sesión reubicada.

Nótese cómo el algoritmo es capaz de compensar casos en los que la capacidad del nodo perdido es mayor a la capacidad individual de cada nodo disponible, encendiendo tantos nodos como sea necesario para recuperar la capacidad anterior al fallo, si éstos están disponibles. Por ejemplo, si se pierde un nodo de 40 sesiones de capacidad, el gestor de tolerancia a fallos encenderá dos nodos de 20 sesiones de capacidad cada uno, si éstos están disponibles (apagados o en descarga).

En el prototipo implementado, todas las operaciones de encendido utilizan el mismo módulo, por lo que el bloqueo de encendido de nuevos nodos es siempre coherente.

Debido a que el procesamiento de la reconfiguración se realiza de forma incremental en cada una de las sesiones que se están reubicando, con el bloqueo de sesiones activo se evita un ciclo de encendidos de todos los nodos disponibles.

El caso de una situación en la que se ha ordenado el encendido de un nodo y, mientras éste se activa, se pierde uno de los nodos ya operativos, el bloqueo de sesiones va a prevenir encender un nodo adicional para suplir la capacidad perdida. Para solucionar este caso particular, es necesario establecer dos bloqueos:

- Un bloqueo para el encendido de nodos en recuperación, de forma que sólo un nodo pueda estar en estado de encendido para suplir capacidad perdida.
- Otro bloqueo para el encendido de nodos en condiciones normales de forma que sólo pueda encenderse un nuevo nodo si ninguno de los dos bloqueos están tomados.

Como puede observarse sólo existe un caso en el que dos nodos puedan estar encendiéndose, y es cuando se pierde un nodo mientras otro estaba en su secuencia de arranque.

4.2.5 Integración con el algoritmo de gestión energética

Este sistema de tolerancia a fallos se encuentra muy intrincado con el algoritmo de gestión energética descrito y validado en [Medrano11].

Todos los mecanismos de gestión de la capacidad y balanceo de carga son los mismos que en dicho algoritmo y el sistema descrito es una aplicación del algoritmo de gestión energética, de forma que se expande al sistema de gestión autonómica a la tolerancia a fallos, acercándose al objetivo de la *self-optimization*.

En el Gráfico 10 se expone el funcionamiento de todo el sistema de tolerancia a fallos. Como se puede observar, el mecanismo de reubicación de sesiones se desencadena en dos puntos concretos del procesamiento de una sesión: cuando no se puede abrir una conexión con uno de los nodos trabajadores o cuando la conexión que ya se tiene abierta se rompe.

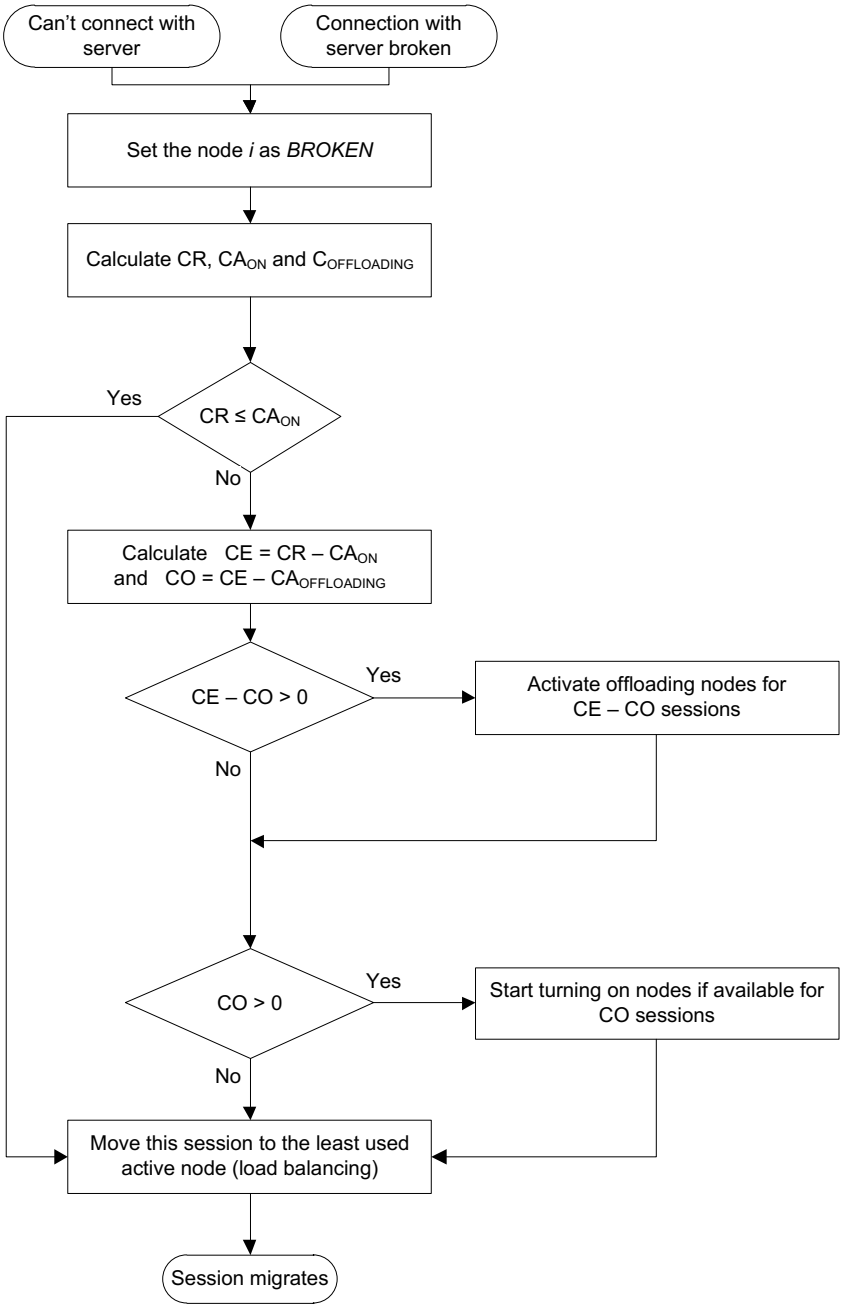

Gráfico 10: Diagrama de flujo del algoritmo de reubicación de sesiones

En cualquiera de los dos casos, la sesión ya ha sido admitida por el módulo de balanceo de carga del gestor autonómico, por lo que no es conveniente romperla (para evitar afectar a la calidad percibida por el cliente).

Una vez que se detecta la rotura del nodo, se marca como inutilizable y se calculan los parámetros pertinentes, tal y como se han detallado en secciones anteriores.

Una nota importante en relación a este diagrama de flujo es que este proceso se aplica para cada una de las sesiones que se encuentran asignadas al nodo estropeado, de esta forma, el algoritmo es totalmente adaptativo a las condiciones cambiantes del clúster, como nuevas roturas, y la distribución de la carga a los demás nodos es más uniforme y progresiva.

Por este motivo, es muy importante mantener las utilizaciones, capacidades y números de sesiones asignadas a los nodos (incluido a los rotos, por las sesiones que están migrando) actualizadas en todo momento para que el algoritmo converja correctamente.

Todos los mecanismos de asignación de sesiones y cálculo de capacidades y utilizaciones son los que el gestor autonómico ya proporciona para la gestión energética.

Finalmente, se produce una migración de sesión y se desencadena el algoritmo de balanceo de carga con gestión energética descrito en la sección 3.3.

5 PRUEBAS Y RESULTADOS

Esta sección resume los mecanismos de validación y prueba aplicados al algoritmo integrado de tolerancia a fallos y las mejoras para clústeres heterogéneos así como los resultados de las mismas en diversas situaciones.

5.1 Modelado de respuesta y energía

Tras modificar, con respecto al proyecto fin de carrera que antecedió a este trabajo fin de máster, el tratamiento de clústeres heterogéneos y ajustar la carga para los experimentos de pruebas, ha sido necesario re-estimar los modelos energéticos que describen el clúster de computadores de pruebas. Además, se ha agregado una cabina de discos, que permite agregar un mayor componente de heterogeneidad al clúster. En concreto, se tienen tres tipos de nodos: con procesador Intel utilizando un disco interno, con procesador Intel utilizando una cabina de discos y con procesador AMD utilizando un disco interno.

5.1.1 Nodo Intel con disco interno

El nodo de tipo Intel, basado en un HP ProLiant DL380 con procesador de microarquitectura Nehalem de Intel y dos procesadores de 4 núcleos cada uno, además de 16 GiB de memoria principal cuenta con una gran capacidad, de 101 usuarios para satisfacer el SLA de 2 segundos, tal y como muestra el Gráfico 12.

Si se observa la evolución de las utilizaciones de dispositivos mostrada en el Gráfico 11, se deduce cómo el disco interno es el dispositivo que alcanza la saturación antes de los demás, de forma que es el dispositivo limitante para la capacidad total del servicio.

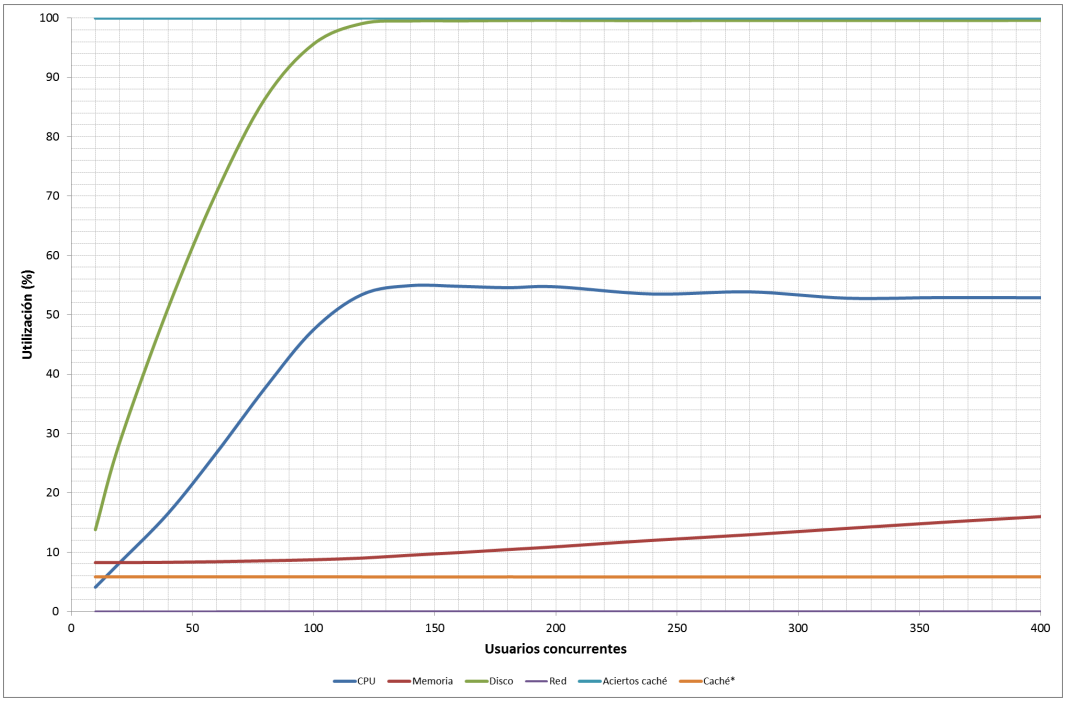

Gráfico 11: Utilizaciones de dispositivos en función de la carga (Intel)

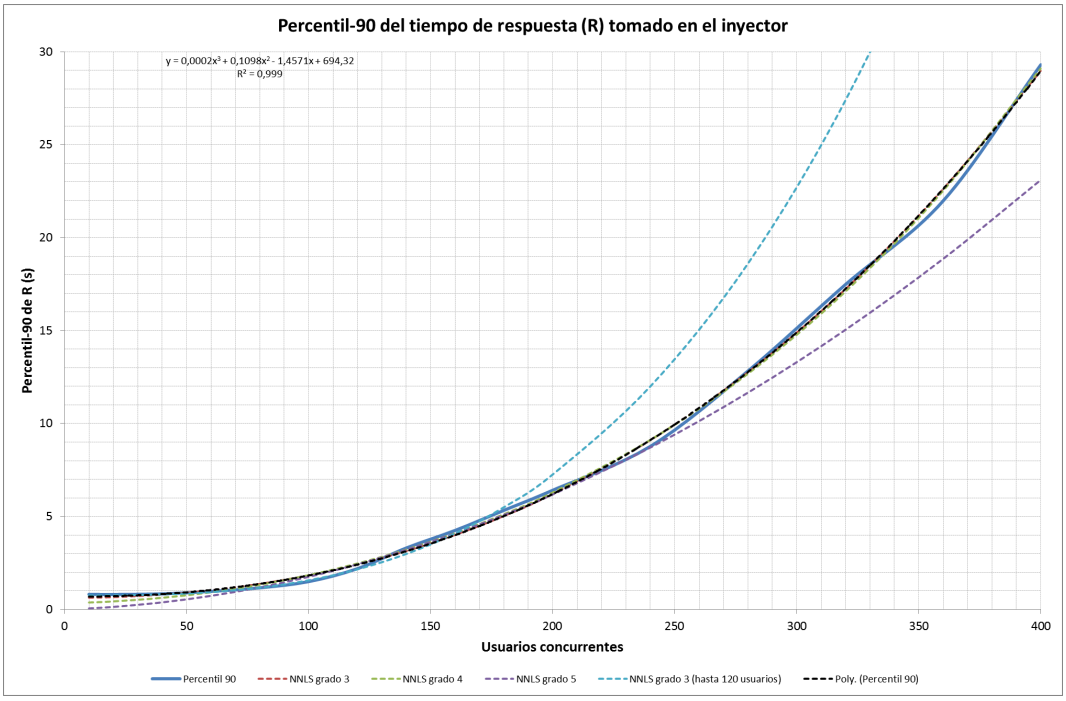

Gráfico 12: Modelos de respuesta del nodo Intel

En lo referente al consumo energético, cuyos resultados se resumen en el Gráfico 13, se observa cómo hasta 140 usuarios el control de frecuencia está operando, teniendo un consumo lineal, mientras que a partir de este punto, se alcanza el consumo máximo y se estabiliza, entrando en zona de eficiencia, donde agregar usuarios al servidor no cuesta más en términos de consumo energético adicional.

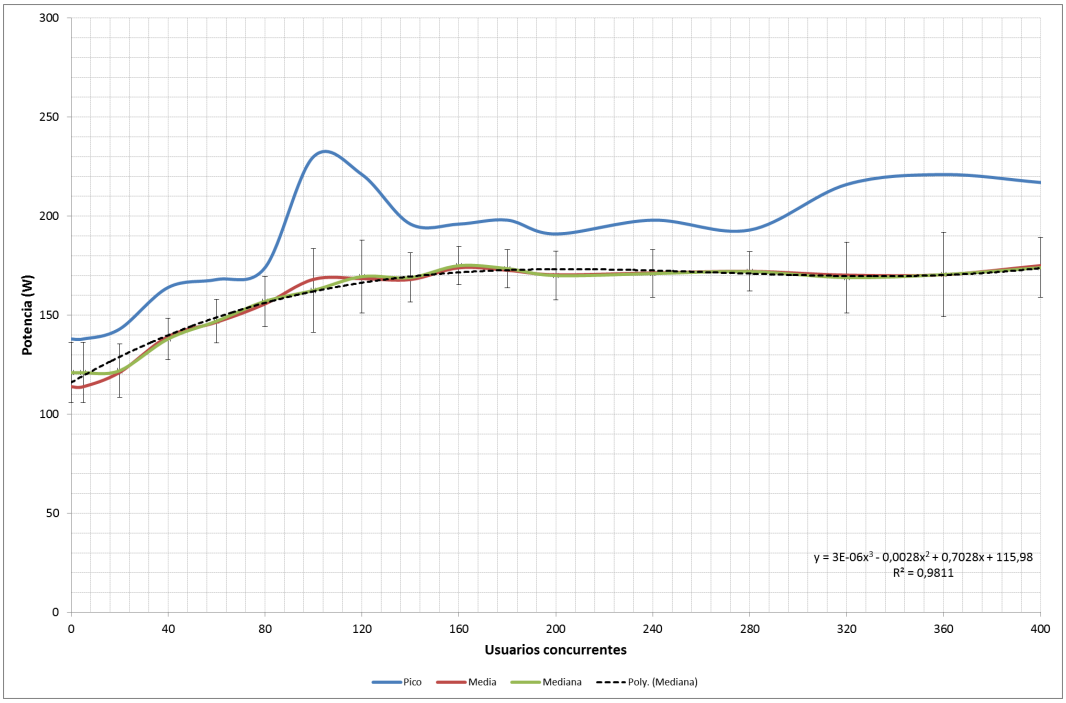

Gráfico 13: Consumo energético en función de la carga (Intel)

5.1.2 Nodo Intel con cabina de discos

Este nodo es el modelo con mayor capacidad del clúster, puesto que además de alta capacidad de CPU, tiene una alta capacidad de disco, en comparación con el nodo Intel sin cabina.

Tal y como se observan en los gráficos de utilización de dispositivos en función de la carga para ambos nodos, este nodo, que es idéntico al nodo Intel, pero está usando el almacenamiento de la cabina en lugar del disco interno, tiene una saturación más alta, en los 153 usuarios, de también la cabina de discos.

La capacidad añadida de la cabina de discos permite tener una utilización más equilibrada del disco y la CPU, estando esta en torno al 90% de uso.

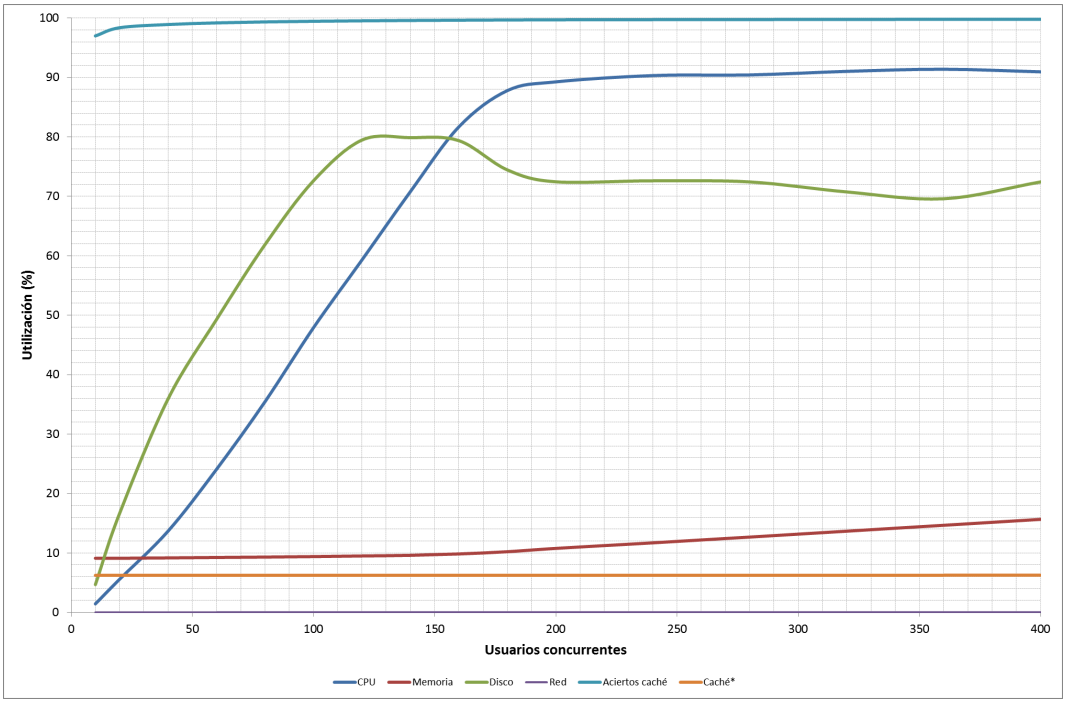

Gráfico 14: Utilizaciones de dispositivos en función de la carga (Intel con cabina)

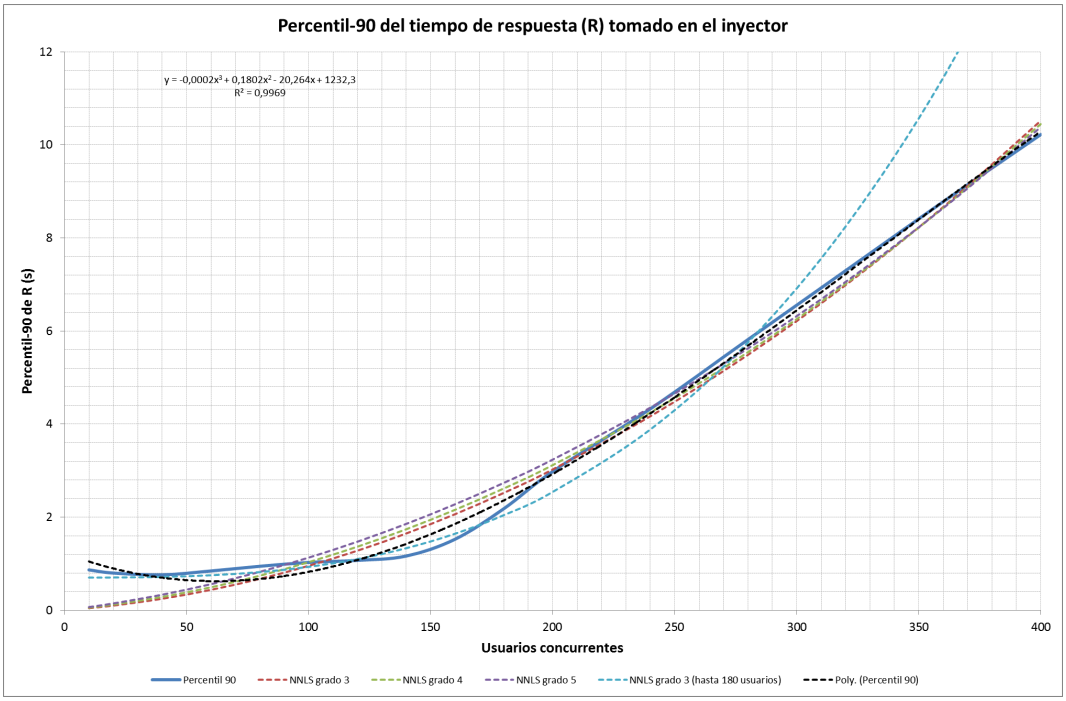

Gráfico 15: Modelos de respuesta del nodo Intel con cabina de discos

En el Gráfico 16 puede observarse que el comportamiento energético de este nodo es muy similar al del nodo Intel sin cabina, pero con un consumo más alto en los rangos de saturación.

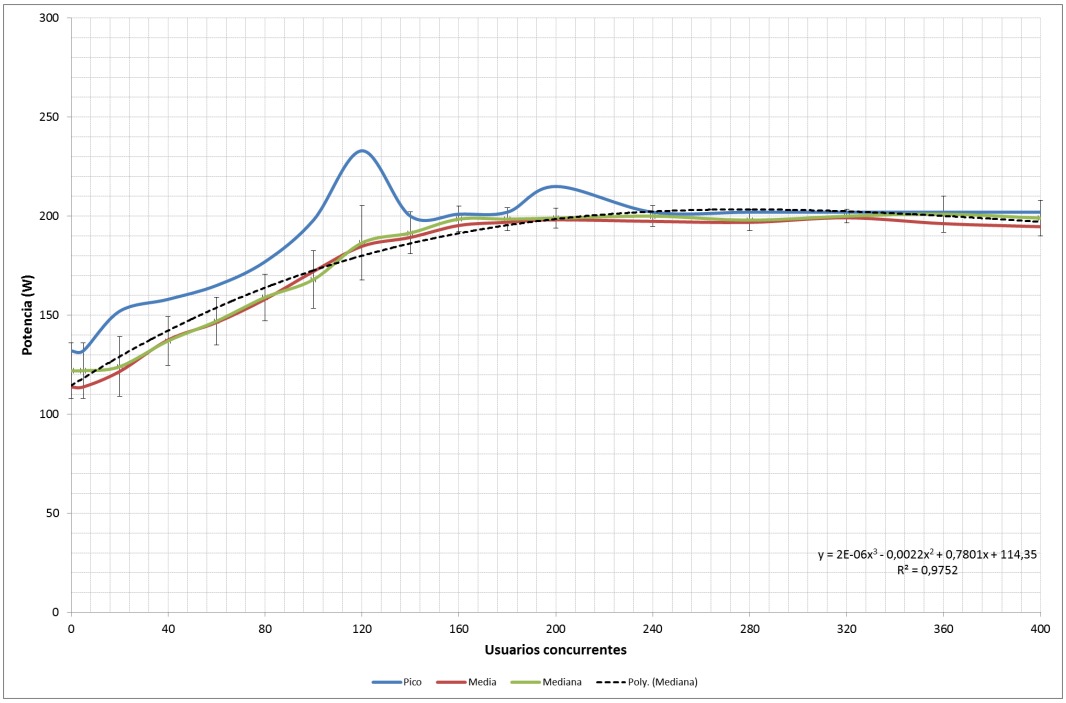

Gráfico 16: Consumo energético en función de la carga (Intel con cabina)

5.1.3 Nodo AMD

El nodo AMD tiene dos procesadores Opteron de dos núcleos cada uno, además de 2 GiB de memoria principal.

En este caso siempre se usa el disco interno y el dispositivo limitante es el procesador. Estos nodos soportan 29 usuarios concurrentes sin violar el SLA de 2 segundos.

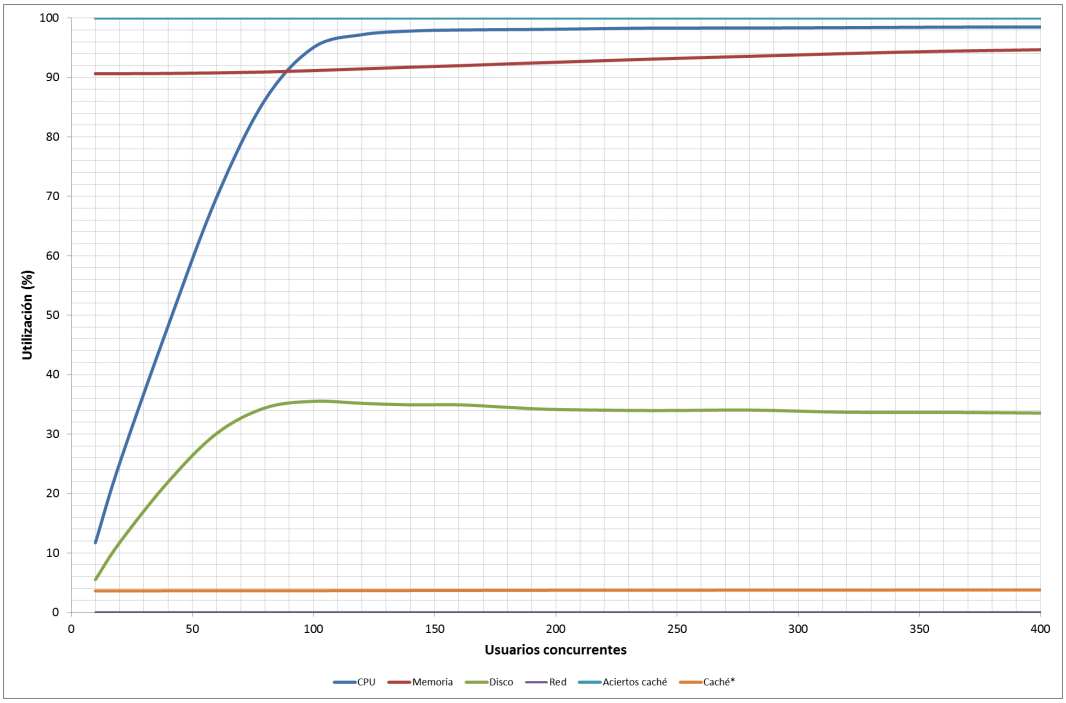

Gráfico 17: Utilizaciones de dispositivos en función de la carga (AMD)

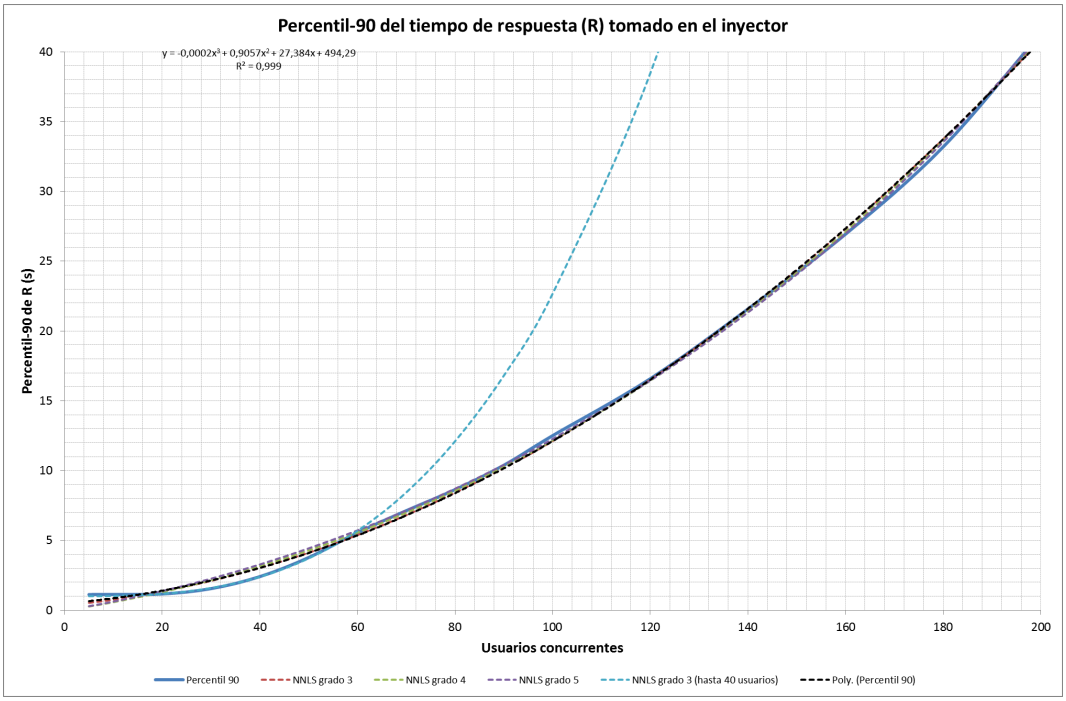

Gráfico 18: Modelos de respuesta del nodo AMD

En los términos de eficiencia energética, se observa como el consumo absoluto de energía es más alto y la saturación del consumo se alcanza antes.

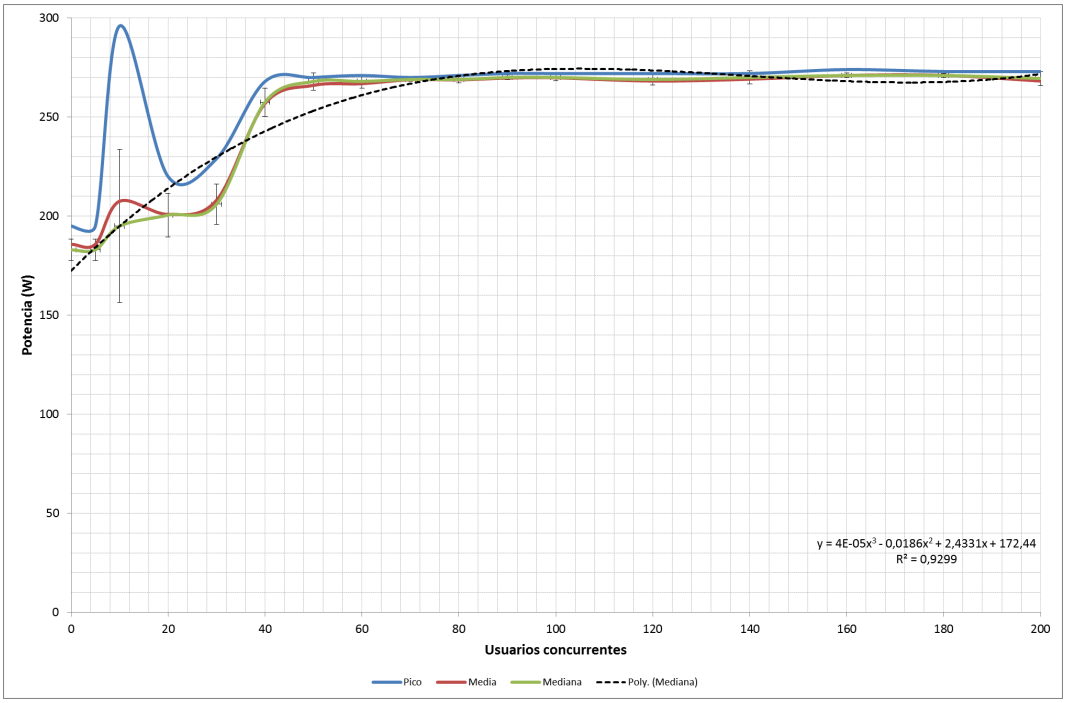

Gráfico 19: Consumo energético en función de la carga (AMD)

5.2 Pruebas de validación en clústeres heterogéneos

Al igual que con el desarrollo original del prototipo de gestión energética, la ampliación para tolerancia a fallos se ha probado usando un mecanismo ágil, basado en pruebas de validación que comprenden varios estados de funcionamiento y pruebas de análisis estático.

Esta sección presenta los resultados de prueba de las mejoras realizadas con respecto al algoritmo presentado en [Medrano11] en el cálculo del umbral de apagado dinámico U_{din}.

Los parámetros usados para estas pruebas son los mismos que en la validación inicial del algoritmo, usando una carga ascendente y descendente que recorre todos los estadios de funcionamiento del clúster, incluyendo la sobresolicitación, que permiten encender toda la capacidad del clúster y posteriormente volver a un estado de consumo mínimo.

Se han usado los parámetros de 0,9 y 0,8 para los umbrales de encendido y post apagado, de forma que el umbral de pre apagado se calcula de forma dinámica con las mejoras explicadas en la sección 3.6.2. Para la captura de información relativa a la calidad de servicio, se ha usado el mecanismo de ventanas de captura, con 100 segundos de intervalo.

En los gráficos resumen, la capacidad de los nodos se tiene en cuenta siempre que está disponible, esto es, cuando termina el encendido se tiene en cuenta y cuando se ordena el apagado se deja de tener en cuenta. Con el consumo energético se realiza el mismo cálculo,

En el Gráfico 20 puede observarse el encendido paulatino de los computadores y cómo, al reducirse la carga, se apagan consecutivamente.

La gestión de los nodos en este clúster heterogéneo ha mejorado, al disponer inicialmente de un nodo que no es el más eficiente y se apaga el nodo Intel que tiene la cabina de discos conectada antes del nodo Intel que no cuenta con cabina de discos.

Tal y como describe el Gráfico 20, el clúster ha sido probado con cuatro nodos, de forma que el primer nodo Intel comienza encendido y el resto de nodos apagados. La carga es ascendente en la primera mitad del experimento y, tras completar el 90% de la capacidad del primer nodo se ordena encender el segundo, correspondiente al nodo Intel que no cuenta con cabina de discos. Este patrón de encendido se repite hasta que la capacidad está completa, por lo que posteriormente se aplica control de admisión para rechazar las sesiones que superan la capacidad máxima. Conforme se reduce la capacidad se apagan los nodos, dejando siempre el clúster en el 80% de utilización tras el apagado, tal y cómo la gestión del umbral de post apagado ha sido diseñada.

Se observa cómo la eficiencia energética depende fundamentalmente de la cantidad de nodos operativos y cómo teniendo más nodos encendidos el consumo *idle* de éstos hace que baje la eficiencia total del clúster. Asimismo, se observa cómo en los instantes previos al encendido y los instantes posteriores al apagado de un nodo, el tiempo de respuesta se incrementa ligeramente, aunque siempre satisfaciendo los requisitos del SLA. Esto se debe a que la orden de encendido se da cuando se llega al 90% de utilización y mientras se completa el encendido se puede llegar a utilizaciones del 95%, de forma que el percentil-90 del tiempo de respuesta se aproxima más al SLA. En el caso del apagado, tras la orden de apagado de uno de los nodos se opera con menos capacidad y esa concentración hace que el tiempo de respuesta se incremente ligeramente.

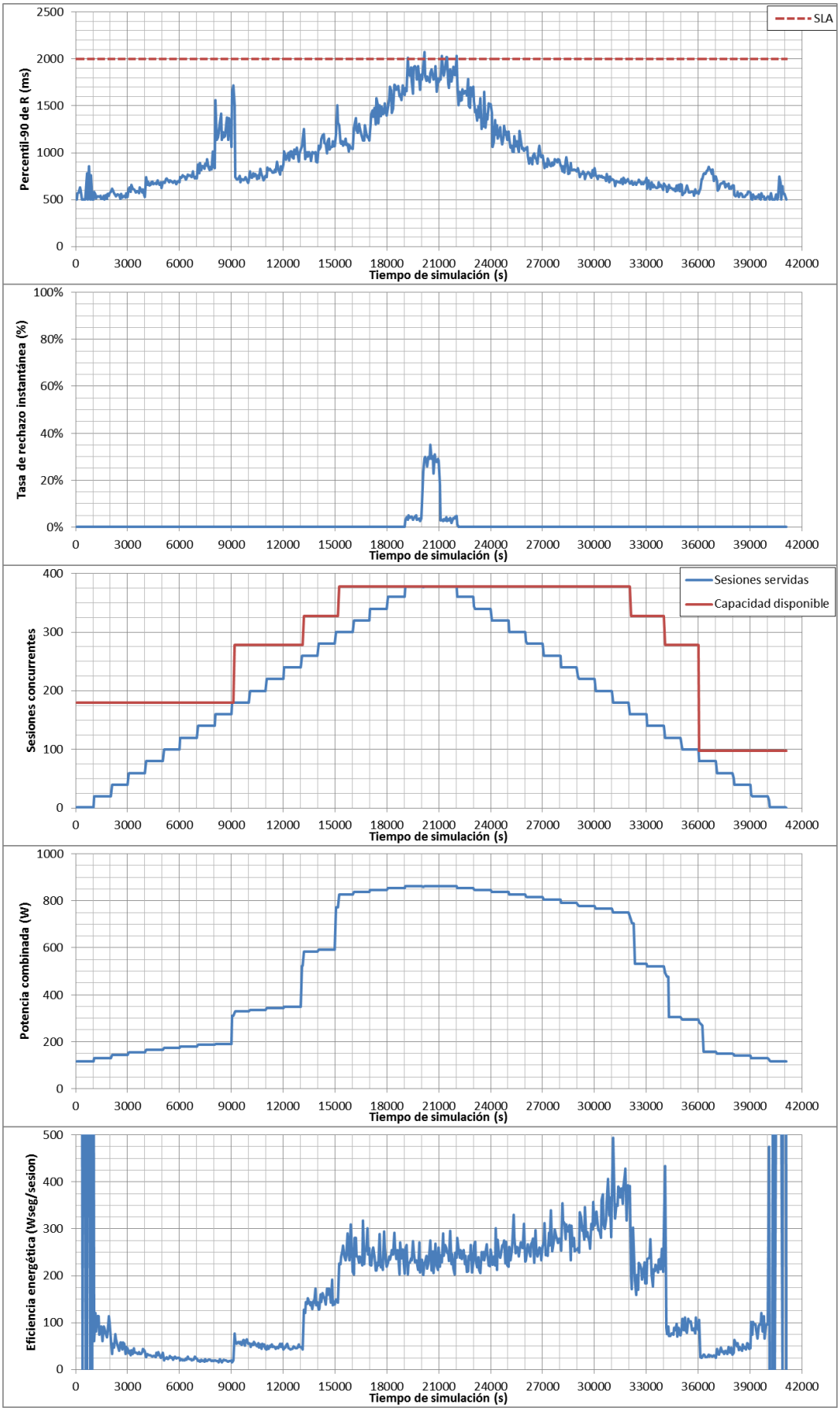

Gráfico 20: Métricas de validación en entornos heterogéneos

5.3 Métricas de violación del SLA

Para el estudio de la tolerancia a fallos, es fundamental medir cómo la aparición de fallos y su corrección influye en la calidad de servicio, en términos de su influencia en el tiempo de respuesta del servicio y en términos de violación del SLA.

En este segundo caso, es deseable obtener dos métricas: una métrica de violación del SLA global de un experimento, que permita comparar situaciones de diversa configuración y una métrica dinámica que permita ver cómo evoluciona un experimento y cuándo se producen las violaciones del SLA.

Teniendo en cuenta que un SLA es un acuerdo entre el proveedor del servicio y un cliente (de forma individual) por el que las peticiones de éste que conformen una sesión no deben superar un tiempo de respuesta determinado (generalmente se aplica un percentil, como el 90%, para relajar ciertas situaciones de peticiones conflictivas u outliers), es necesario que estas métricas se calculen de forma que sean significativas para el cliente, que en definitiva es el elemento del sistema que va a percibir la respuesta del clúster como una caja negra, sin importancia de la infraestructura subyacente o fallos internos del servicio.

Fundamentalmente, pueden proponerse dos clases de métricas, cada una con sus objetivos, ventajas e inconvenientes:

- Métricas dinámicas, donde se mide el incumplimiento del SLA de forma dinámica. Permiten, mediante series temporales, observar la evolución de un experimento individual. El uso de ventanas para el cálculo de la serie temporal es recomendable para el alisamiento y amortiguación de los datos.
- Métricas resumen, que comprenden todos los datos y resultan en una métrica numérica para permitir comparar experimentos en diversas situaciones.

5.3.1 Métricas basadas en ventanas

Estas métricas, introducidas en [Garcia09], son muy apropiadas para observar la evolución de un experimento:

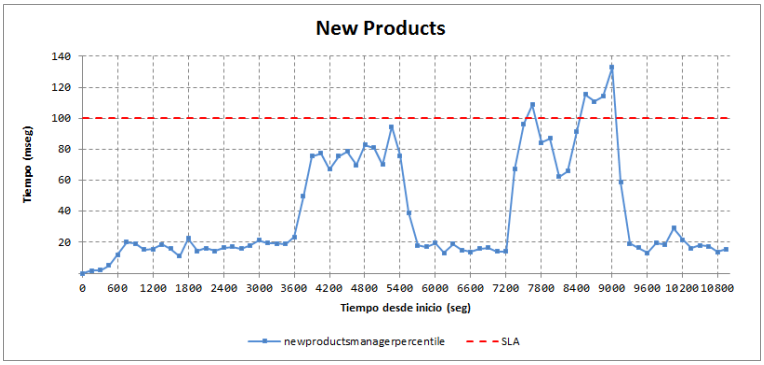

Gráfico 21: Ejemplo de métrica de cumplimiento del SLA dinámica

Tal y como se observa en el Gráfico 21, una métrica dinámica, es decir, una métrica calculada a lo largo de todo el experimento de forma instantánea, es muy útil para comprobar dónde se producen violaciones y aislar los motivos de las mismas. El cálculo basado en ventanas permite eliminar outliers y tener una evolución más suave y realista del experimento.

En concreto, el cálculo de la métrica mostrada anteriormente se basa en tener una cola circular donde se van registrando los datos relativos a la métrica a componer, en este caso el tiempo de respuesta de todas las transacciones que se estén procesando. Cada cierto tiempo prefijado, el intervalo de cálculo de las métricas de QoS, se accede a la cola circular para obtener los resultados almacenados correspondientes a los últimos n segundos y se compone la métrica, en este caso con el percentil 90. De este modo, el cálculo de las métricas dinámicas se basa en una ventana deslizante.

5.3.2 Métricas resumen

El objetivo de las métricas resumen es más concreto que el de las métricas dinámicas, se concentra en ofrecer un resultado numérico que permita comparar dos experimentos de forma sencilla.

Una primera métrica que permite satisfacer las restricciones establecidas en la definición de SLA es la que sigue:

$$Métrica_1 = \frac{nº\ de\ sesiones\ que\ incumplen\ el\ SLA}{nº\ total\ de\ sesiones} \cdot 100$$

Esta métrica es completa en términos de satisfacción de las restricciones de la definición del SLA, pero presenta un problema importante: la cuantización que sufre cuando las sesiones no tienen muchas transacciones.

Para calcular si una sesión ha incumplido el SLA, se debe de calcular el 90-percentil del tiempo de respuesta de las transacciones que han compuesto dicha sesión, de forma que si supera el SLA, se cuenta como una sesión que lo ha incumplido.

Si se tiene una media de unas 10 transacciones por sesión, por ejemplo, esta métrica se vuelve muy dependiente de una sola transacción, lo que hace que oscile demasiado. Por esto, se puede redefinir de forma que el elemento básico sea la ventana:

$$Métrica_2 = \frac{nº\ de\ ventanas\ que\ incumplen\ el\ SLA}{nº\ total\ de\ ventanas} \cdot 100$$

Aunque de esta forma no se satisface de forma plena la definición de SLA, se reduce mucho el efecto de cuantización y se hace independiente de la duración de la sesión. De esta forma, todas las métricas dinámicas se reducen a métricas basadas en ventanas.

Debido a que cada ventana puede contener, en un clúster medio con una utilización plena, miles de transacciones, la fiabilidad de esta métrica es equivalente a la anterior.

El cálculo de una métrica resumen se puede basar en la composición de una métrica instantánea representada en forma de clústeres de puntos en la figura siguiente, que representa las violaciones del SLA para un usuario. Si éstos clústeres de puntos de combinan, se puede obtener una métrica resumen compuesta por todas las violaciones de sesión de todos los usuarios servidos.

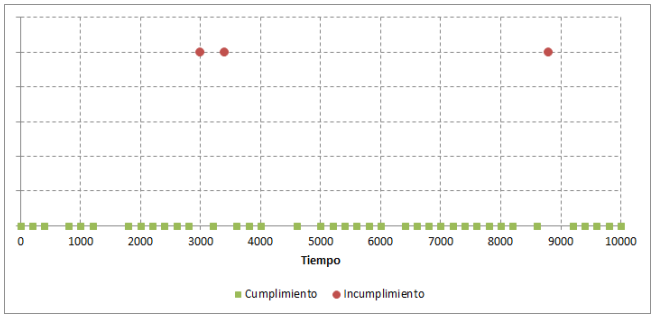

Gráfico 22: Representación en clústeres de una métrica resumen

Fundamentalmente, el objetivo de las métricas resumen es obtener una forma sencilla de comparación de experimentos, por lo que no se mostrará una evolución de su cálculo.

5.4 Pruebas de tolerancia a fallos

Las pruebas de tolerancia a fallos permiten medir la influencia de fallos en diversas situaciones de carga y cómo el gestor autonómico es capaz de recuperar la capacidad y la calidad de servicio. La relación entre violación del SLA y el fallo de los nodos está también entre los objetivos del estudio.

5.4.1 Diseño experimental y objetivos del estudio

A continuación se detallan las condiciones de carga en las que se van a realizar experimentos individuales para comprobar el correcto funcionamiento y la eficacia del algoritmo en diversas condiciones:

Situación	Tipo de carga	Característica a estudiar
Media carga	Estacionaria	Reasignación de sesiones
Alta utilización	Estacionaria	Encendido de nodos
Sobresolicitación	Estacionaria	Control de admisión
Carga ascendente	Dinámica	Adaptación de capacidad
Carga descendente	Dinámica	Rescate de nodos en descarga

Tabla 2: Diseño experimental de las pruebas

En todos los experimentos se realizarán pruebas en las que falla un nodo para comprobar la dependencia respecto al cumplimiento del SLA. La configuración experimental del clúster es la mostrada en el Gráfico 23.

Se han usado los nodos AMD y el nodo Intel con cabina de discos, debido a una avería en el otro nodo Intel. La configuración de la captura de información de calidad de servicio es la misma que anteriormente, usando el método de ventanas con 100 segundos de intervalo de cálculo, que determina la resolución de los gráficos que se presentan a continuación.

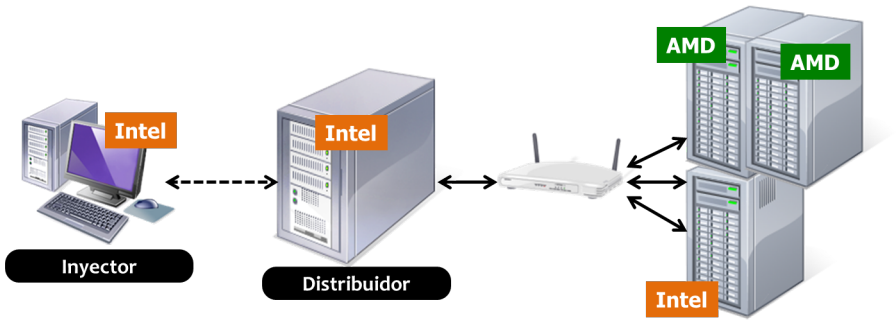

Gráfico 23: Configuración del clúster de pruebas

Los nodos basados en procesador AMD presentan una capacidad de 29 sesiones, mientras que el nodo basado en procesador Intel, soporta 153 sesiones concurrentes. Éste cuenta con una cabina de discos que permite incrementar las prestaciones del sistema de entrada/salida.

El inyector se encuentra instrumentado para registrar toda la información relativa a calidad de servicio percibida, asimismo el repartidor de carga (distribuidor) está recolectando información de calidad de servicio mediante la técnica del muestreo de ventanas, con medidas cada 100 s [Garcia09].

Es necesario recordar en este punto cómo no es posible que se den ciertas situaciones limitadas por el gestor energético, por ejemplo, tener una configuración de clúster con varios nodos encendidos a una baja utilización o nodos con una carga muy diferente[1].

5.4.2 Período transitorio de adaptación

Es necesario analizar el comportamiento del algoritmo de tolerancia a fallos durante el periodo transitorio en que los nodos pueden estar funcionando con una utilización mayor a la que garantiza el SLA, esto es, con $U_i > 1$.

Este período transitorio se define como el intervalo de tiempo en el que el clúster está operando de forma no estable mientras se recupera de un fallo de uno de los nodos. Concretamente, este intervalo de tiempo comienza con la detección del fallo y termina cuando se cumplen las siguientes dos condiciones:

[1] Podrían tenerse nodos muy descargados si se usan valores de $U_{mín}$ muy bajos, pero carece de sentido debido a la pérdida de la optimización energética.

1. Si no es necesario encender máquinas adicionales para absorber la capacidad perdida, el transitorio concluye cuando la última sesión asignada al nodo que ha fallado continua su ejecución en uno de los nodos operativos. Nótese como en esta situación no va a haber sobresolicitación después de la rotura.
2. Si hay que activar o encender algún nodo adicional, el transitorio concluye cuando, además de todas las sesiones del nodo perdido están migradas a otro nodo, todas las máquinas cuentan con una utilización igual o inferior a 1,0. Este caso incluye la adaptación a la nueva capacidad máxima en el caso de situaciones de sobresolicitación y situaciones de espera por el encendido de nuevos nodos.

En el momento en el que uno de los nodos falla, pueden existir hasta $C - C'$ sesiones adicionales, siendo C y C' las capacidades antes y después del fallo.

Un aspecto muy importante de este régimen de funcionamiento transitorio es la duración temporal que tiene. Por supuesto, dependerá de la forma de la carga y de las prestaciones del clúster, en particular:

- De la cantidad de transacciones por sesión, de manera que a sesiones más largas, más tiempo transitorio.
- De la capacidad del nodo caído, de forma que a mayor capacidad perdida, más tiempo de recuperación.

En una situación de sobresolicitación, o donde la nueva capacidad es inferior a la demanda, el control de admisión es clave, pues con la nueva capacidad, se producirán rechazos de sesión para garantizar el cumplimiento del SLA tras el tiempo transitorio de balanceo de carga.

En cada uno de los experimentos individuales se realizará el análisis de los resultados del mismo, destacando los casos particulares y peculiaridades de cada uno.

5.4.3 Comportamiento en condiciones de carga estacionaria media

En situaciones de una utilización media del clúster, éste estará funcionando en el entorno de la mitad de toda la capacidad posible, pero con sólo un subconjunto de nodos encendidos. Éstos estarán operando con una utilización entre el 70% y el 90%.

En concreto el nodo Intel y uno de los nodos AMD estarán operando, manteniendo un nodo AMD apagado. El nodo que va a fallar en este experimento es el nodo AMD.

Cuando se produzca el fallo de uno de los nodos activos, otro de los que se encuentran apagados suplirá la capacidad que estaba soportando el nodo caído. Es necesario tener en cuenta que, dependiendo de los estados de hibernación que se consideren, el tiempo de recuperación del nodo que va a ser activado puede ser mayor o menor.

En las pruebas cuyos resultados se resumen en el Gráfico 24 y el Gráfico 25, el clúster está operando con una carga estacionaria que supone, aproximadamente, la mitad de su capacidad total. En dicha prueba, de 6 horas (21 600 segundos), se produce un fallo de uno de los nodos hacia la mitad del experimento (11 000 segundos).

El clúster está funcionando de forma estable con dos de los nodos operativos y el otro apagado, ya que con 150 clientes concurrentes los dos nodos operativos están operando al 81% y no hay necesidad de encender un nuevo nodo ($U_{máx} = 0{,}95$) ni apagar ninguno ($U_{mín} = 0{,}9$).[2]

Como se puede observar, en el segundo 11 000 del experimento se producen tres efectos con el fallo de uno de los nodos:

- Se produce un pequeño pico en el tiempo de respuesta. Al quedar el clúster funcionando con un solo nodo mientras se completa el encendido del nuevo nodo, este nodo está funcionando al 94% de su capacidad individual, de forma que, aunque se cumple el SLA, la carga es mayor y, por tanto, el tiempo de respuesta se incrementa.
- La capacidad total baja mientras se enciende el nuevo nodo, como cabía esperar. Asimismo, el consumo del clúster disminuye mientras se espera por el encendido (no obstante, se tiene un nodo en espera de encendido, que también están consumiendo su energía, pero su capacidad no está disponible) y, colateralmente, la eficiencia energética se incrementa. Que la eficiencia mejore es debido a que cuantos menos nodos se tengan encendidos, el consumo básico (*idle*) es notablemente menor (de ahí que la aplicación de técnicas DVFS tenga un efecto limitado).
- La productividad, la tasa de rechazo y la tasa de violación del SLA no varían, debido a que no se supera la capacidad máxima del clúster en ningún momento.

[2] Los valores de los parámetros de operación del algoritmo son los optimizados en [Medrano11]. Es necesario recordar cómo $U_{mín}$ es el valor de utilización de post-apagado.

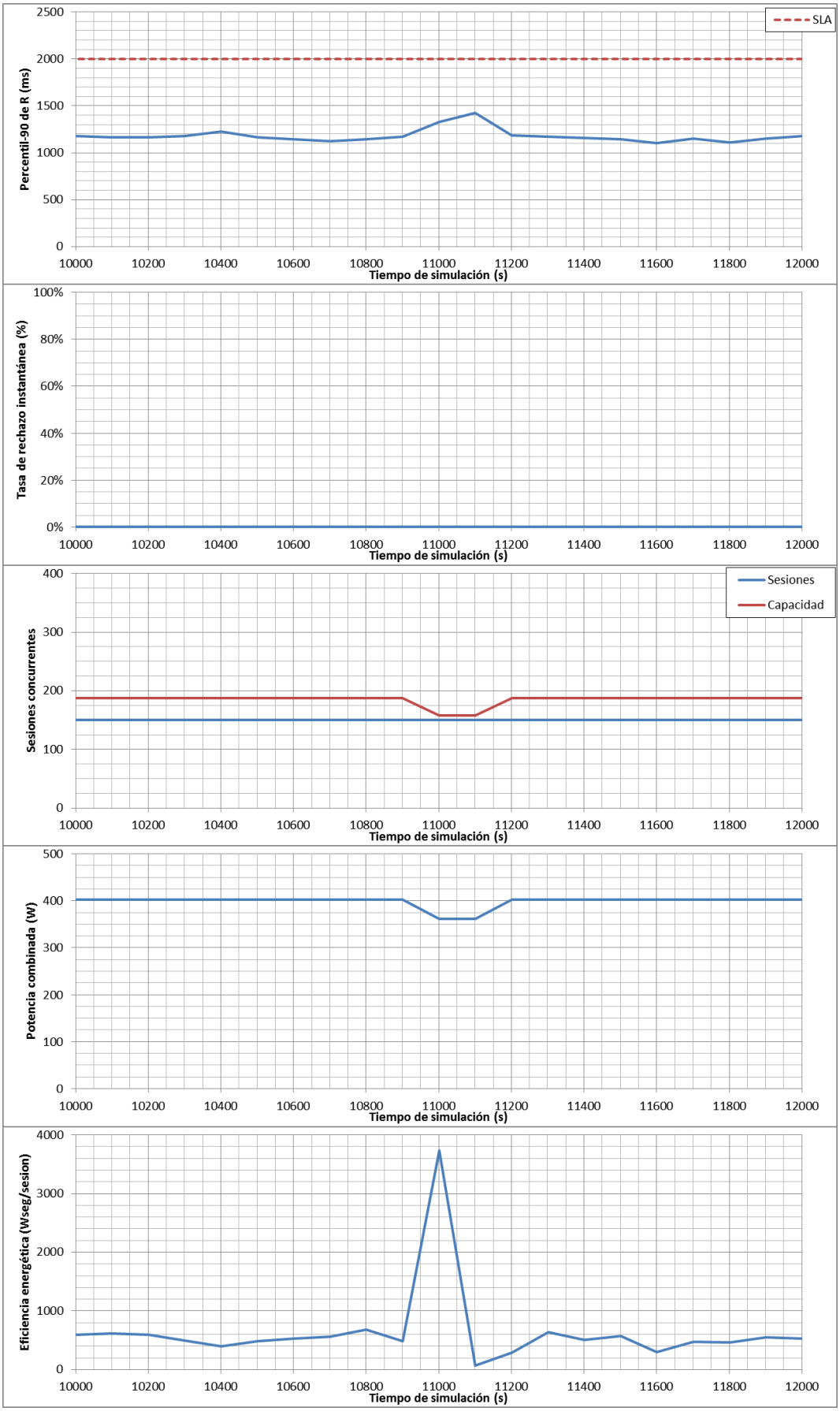

Gráfico 24: Métricas de prueba en media carga (I)

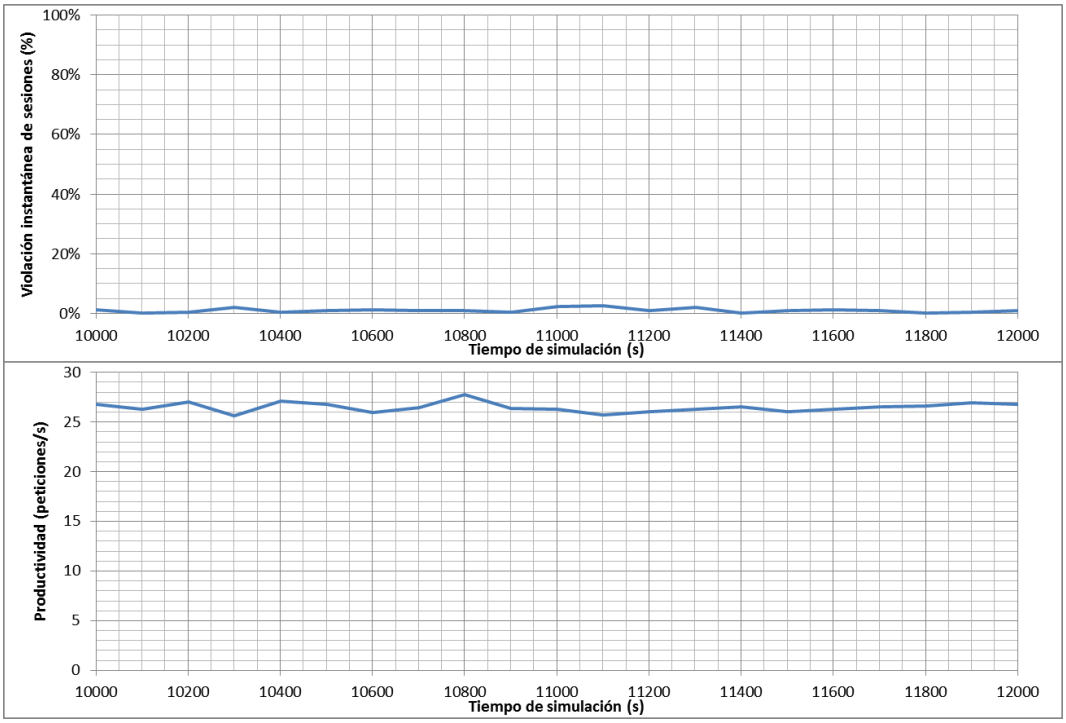

Gráfico 25: Métricas de prueba en media carga (II)

La principal conclusión de este experimento es observar cómo el gestor de tolerancia a fallos responde a la detección de la rotura de uno de los nodos ordenando encender un nuevo nodo de forma inmediata y cómo es necesario esperar un intervalo de tiempo para tener esta capacidad utilizable en el clúster hasta que este nuevo nodo ha completado su secuencia de arranque.

5.4.3.1 Análisis del período transitorio

En este experimento, tal y como y como se muestra en el Gráfico 26, se puede analizar el transitorio que se produce cuando se está operando en condiciones de utilización media, esto es, existen nodos disponibles para encender cuando se produce el fallo de uno de los nodos activos. El Gráfico 26 muestra un detalle del gráfico de capacidad mostrado en el Gráfico 24, pero utilizando datos obtenidos de una medición por eventos en lugar de por muestreo, con lo que la forma obtenida es distinta.

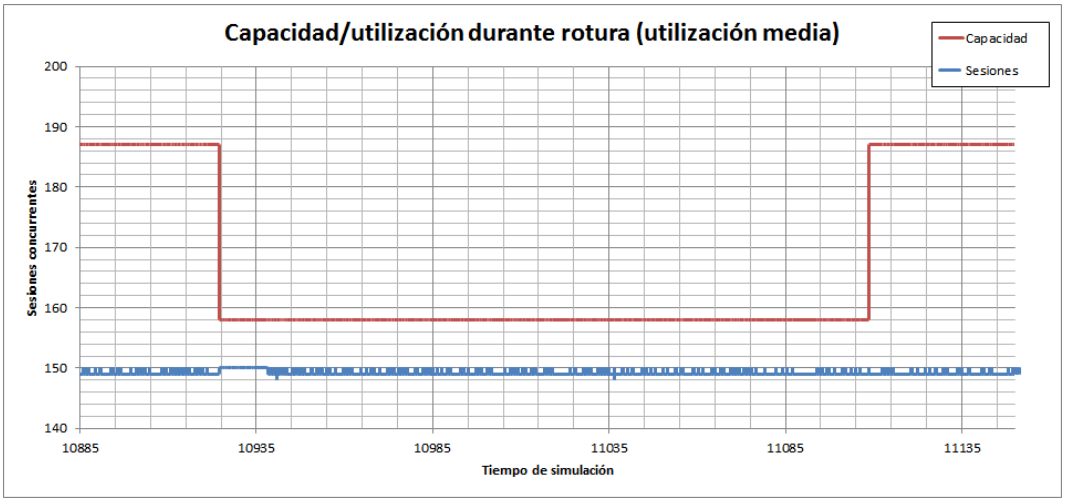

Gráfico 26: Detalle de utilización durante una rotura en utilización media

En este caso, se observa cómo la capacidad requerida por los clientes del clúster permanece constante, mientras que la capacidad disponible se reduce. En este caso, el gestor de tolerancia a fallos ha decido activar uno de los nodos que están apagados, por lo que aparece un tiempo transitorio en el que el clúster está operando de forma no estable a la espera del completado de la secuencia de arranque del nuevo nodo.

Tras la activación extraordinaria del nodo, el clúster vuelve a operar normalmente y a garantizar los requisitos de calidad de servicio, por lo que tiempo transitorio termina. Hay que hacer notar cómo en este experimento sí se cumplen los requisitos de calidad de servicio, pero no está garantizado que ocurra en otros casos ya que los nodos pueden llegar a operar por encima de la capacidad máxima de forma transitoria.

5.4.4 Comportamiento en condiciones de carga estacionaria alta

Una situación de alta utilización es aquella en la que, no estando todos los recursos del clúster en uso, al perder uno de los nodos no existe otro que lo supla y la capacidad total del clúster se reduce.

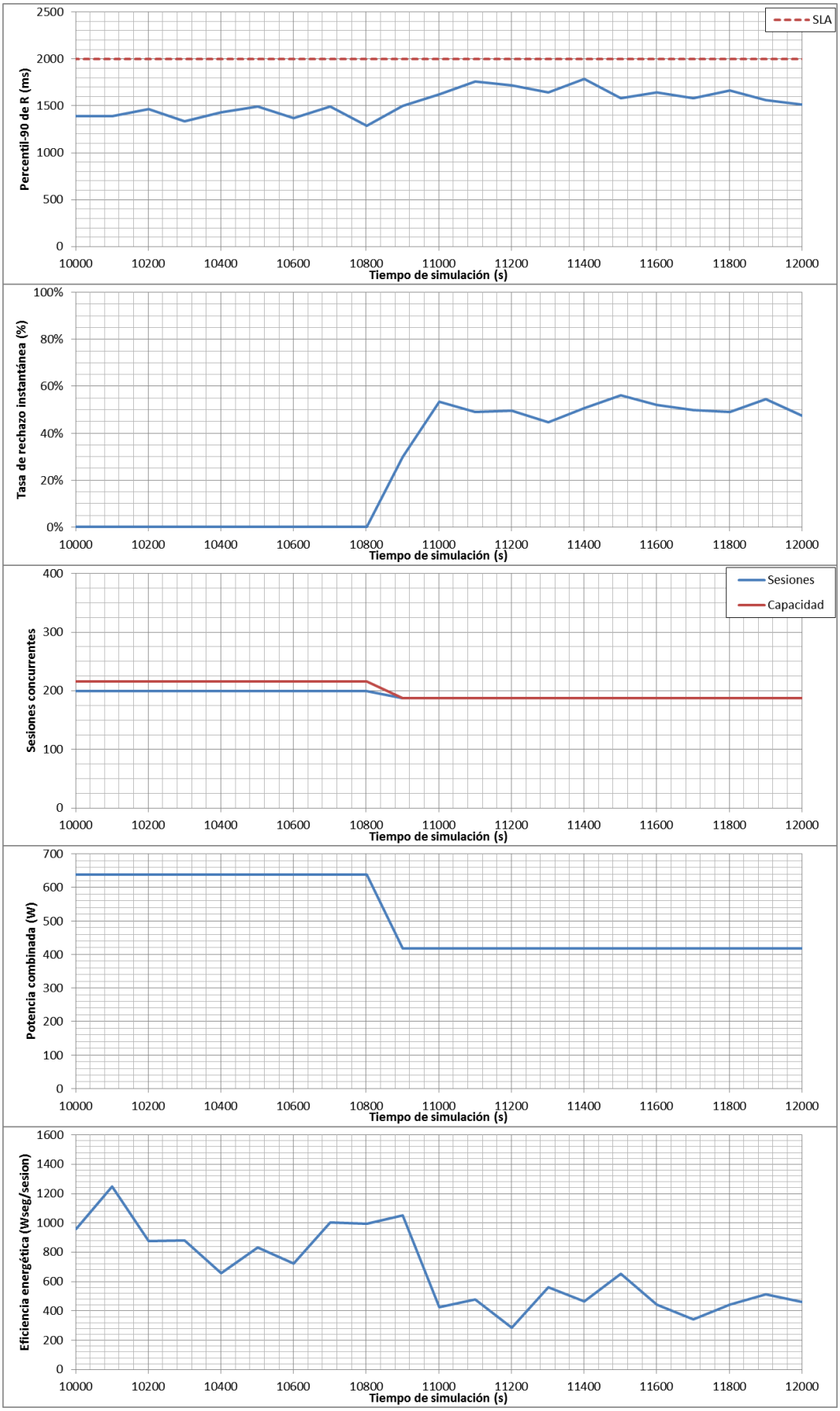

Gráfico 27: Métricas de prueba en alta utilización (I)

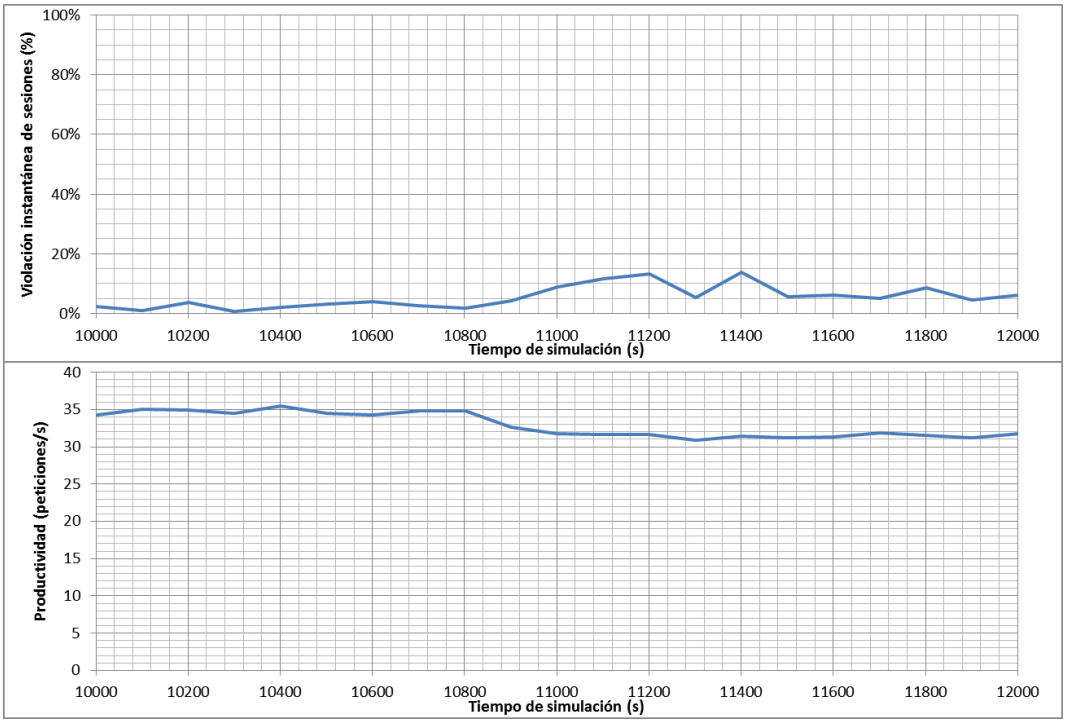

Gráfico 28: Métricas de prueba en alta utilización (II)

En el Gráfico 27 y en el Gráfico 28 se muestran las métricas correspondientes a las pruebas en alta utilización. La configuración de la prueba es idéntica a la mostrada en la sección 5.4.3, a excepción de la carga, esta vez de 200 clientes (el 93% de la capacidad total).

El clúster funciona con todos los nodos cerca de la utilización completa, de forma que el tiempo de respuesta está cercano al SLA, pero satisfaciéndolo, y tiene una oscilación más notable dada la mayor carga en dispositivos menos predecibles, como los discos.

Como en casos anteriores, en el segundo 11 000 de simulación, uno de los nodos falla y el gestor de tolerancia a fallos ha de responder. En este caso, al no haber nodos disponibles, la capacidad del clúster se reduce. Como ésta se reduce por debajo de la utilización actual (y de la demanda de 200 sesiones concurrentes), se produce un proceso de rebalanceo, donde en un periodo transitorio existen más sesiones de las que la capacidad del clúster soporta, ya que son sesiones ya aceptadas y, conforme este período termina, se procede al rechazo de las nuevas sesiones que supondrán una superación de la nueva capacidad.

A partir del fallo en el segundo 11 000 se observa cómo, además de la capacidad reducida, se incrementa la eficiencia al reducirse el consumo, pero se dispara la tasa de rechazos, así como la de violación instantánea de sesiones.

La productividad desciende (al tener menos clientes concurrentemente) y el tiempo de respuesta se incrementa ligeramente, así como se hace también más variable.

Debido a la rotura del nodo, el clúster, para la misma demanda ha cambiado de régimen de operación: de estar en un régimen aceptable de uso (93%) ha pasado a estar en sobresolicitación y la calidad de servicio se ha perjudicado en el sentido de que no se pueden atender a todos los clientes que solicitan servicio.

5.4.4.1 Análisis del período transitorio

Tal y como se muestra en el Gráfico 29, que muestra con detalle el período transitorio de un clúster funcionando en las condiciones de alta utilización de este experimento, que pasa a estar en sobresolicitación al fallar un nodo, existe un período transitorio donde se tienen más sesiones activas en el clúster que capacidad disponible para garantizar el SLA. Cuando estas sesiones terminan (no deben ser terminadas de forma abrupta por el clúster, pues ya han sido admitidas), el clúster funciona a la máxima capacidad de servicio mediante control de admisión.

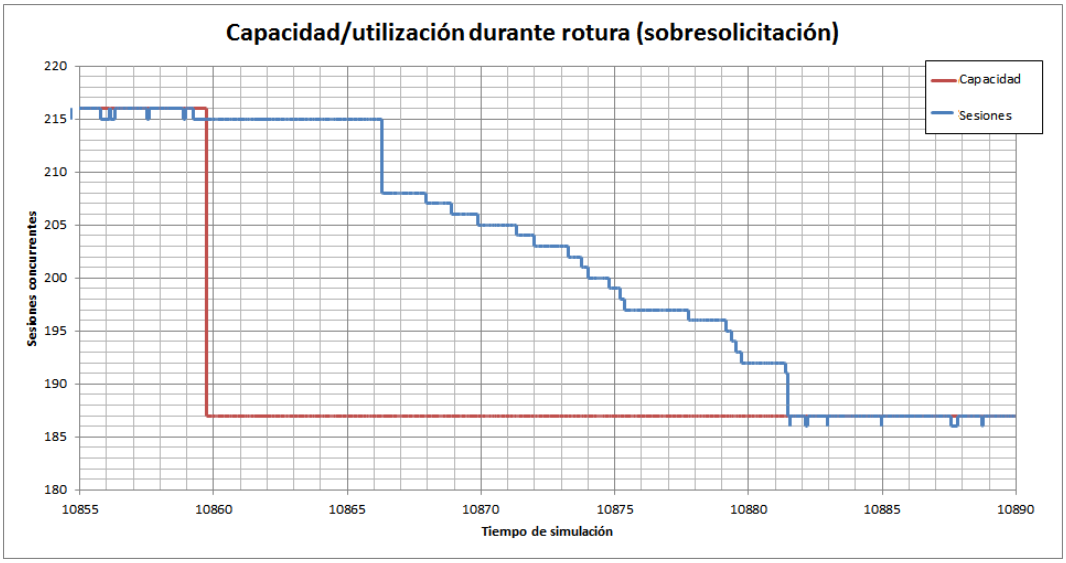

Gráfico 29: Detalle de utilización durante una rotura en alta utilización

Como se observa, el período transitorio dura 37 segundos, entre los 10 927 segundos, momento en el que se produce la rotura, y los 10 960 segundos, momento en el que todas las sesiones están reubicadas y la capacidad del clúster reconfigurada. A partir de este momento, el clúster está funcionando de forma estable de nuevo, aplicando control de admisión para garantizar la calidad de servicio, pues la nueva capacidad es de 187 usuarios y existe una demanda de 200.

5.4.5 Comportamiento en condiciones de sobresolicitación

Una situación de sobresolicitación implica que el clúster soporta una demanda de apertura de sesiones mayor que la que puede satisfacer garantizando la calidad de servicio y el repartidor de carga va aplicar control de admisión para rechazar el servicio a determinados clientes.

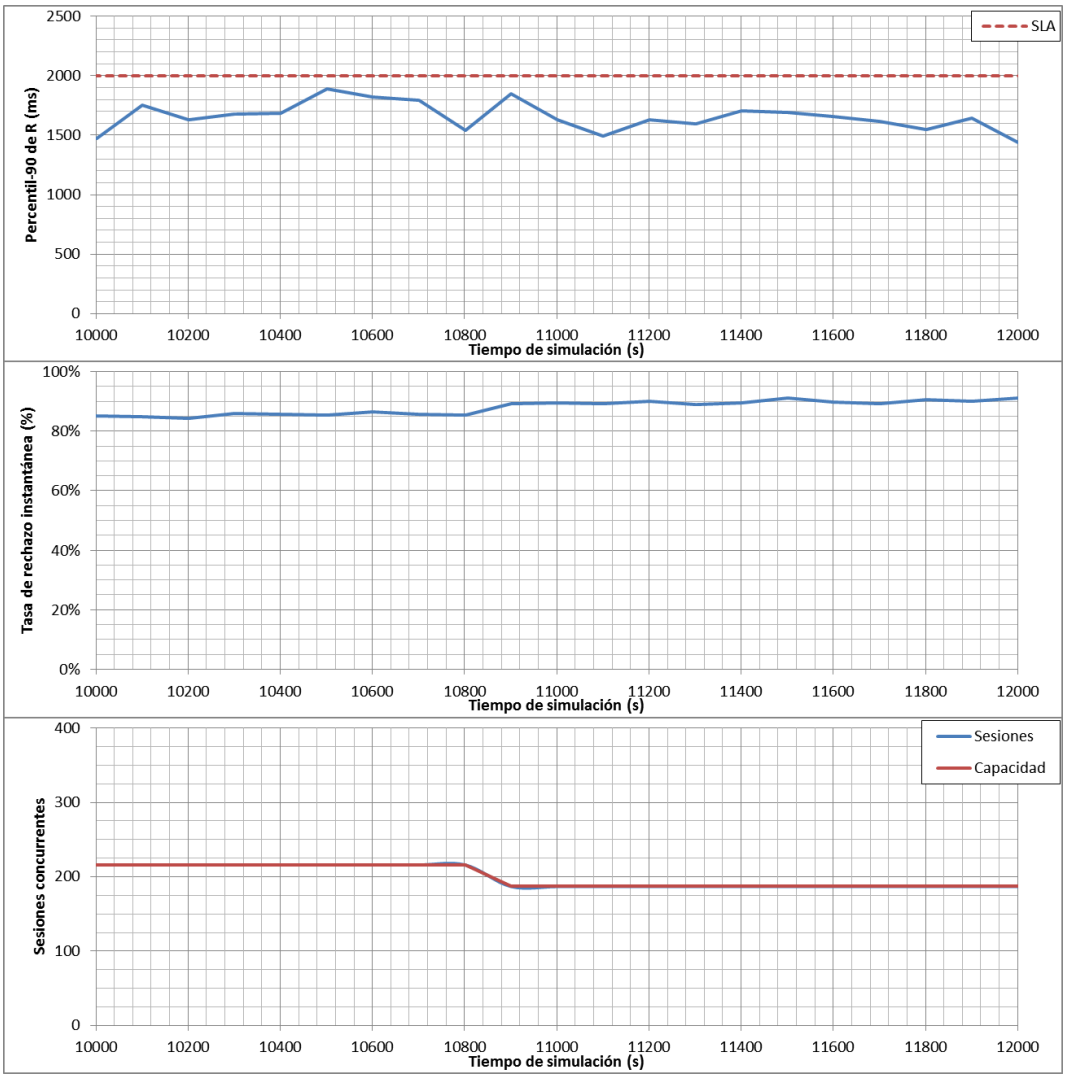

Gráfico 30: Métricas de prueba en sobresolicitación (I)

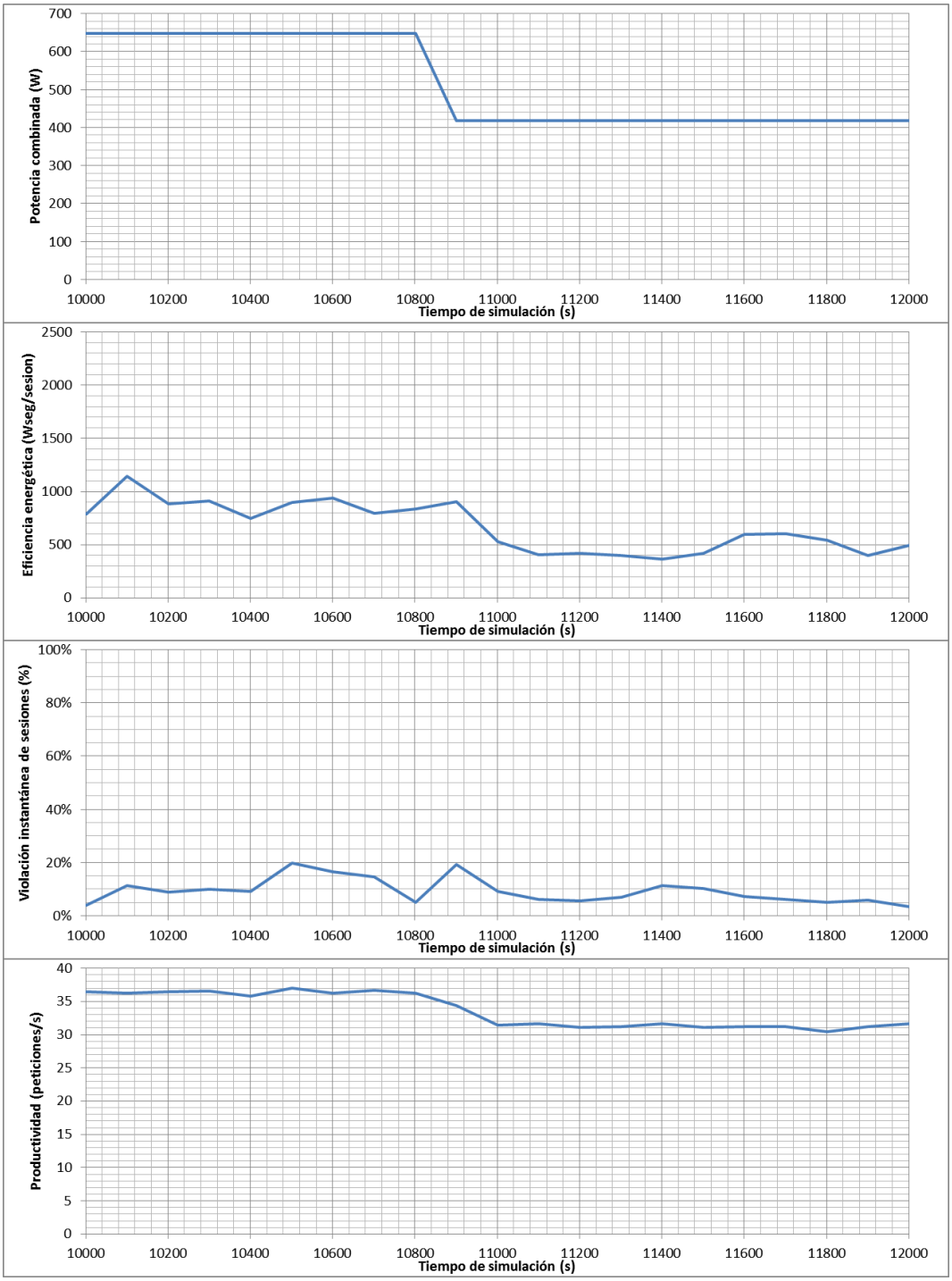

Gráfico 31: Métricas de prueba en sobresolicitación (II)

Tal y como se puede observar, al estar en un régimen de sobresolicitación, siempre existe una tasa alta de rechazos de sesión. Adicionalmente, cuando el nodo se cae en el segundo 11 000 del experimento, ésta se incrementa.

Un aspecto importante de este experimento es comprobar cómo el tiempo de respuesta y la tasa de violación de sesiones permanece invariable. En ambos regímenes de funcionamiento, cada

nodo individualmente está operando al 100% de su capacidad, de forma que cuando uno de ellos se rompe, transitoriamente deberán operar a una capacidad ligeramente superior a su capacidad máxima (como así ocurre entre los segundos 11 000 y 11 030), pero enseguida recuperarán el 100% de utilización.

Lo que sí presenta una variación mayor es la capacidad total y la productividad. Al reducir la capacidad total, se produce un fuerte impacto en la productividad total del servicio (*throughput*).

5.4.5.1 Análisis del período transitorio

El comportamiento del gestor de tolerancia a fallos en estas condiciones de carga alta se detalla en el Gráfico 32. En este caso, se da una situación en la que el clúster está operando con un nivel de carga estacionario mayor que la capacidad del servicio, por lo que se está aplicando control de admisión.

Al fallar uno de los nodos comienza un período transitorio para readaptar la capacidad del clúster, rechazando todas las nuevas sesiones hasta que concluyen las existentes y se normaliza la capacidad utilizada.

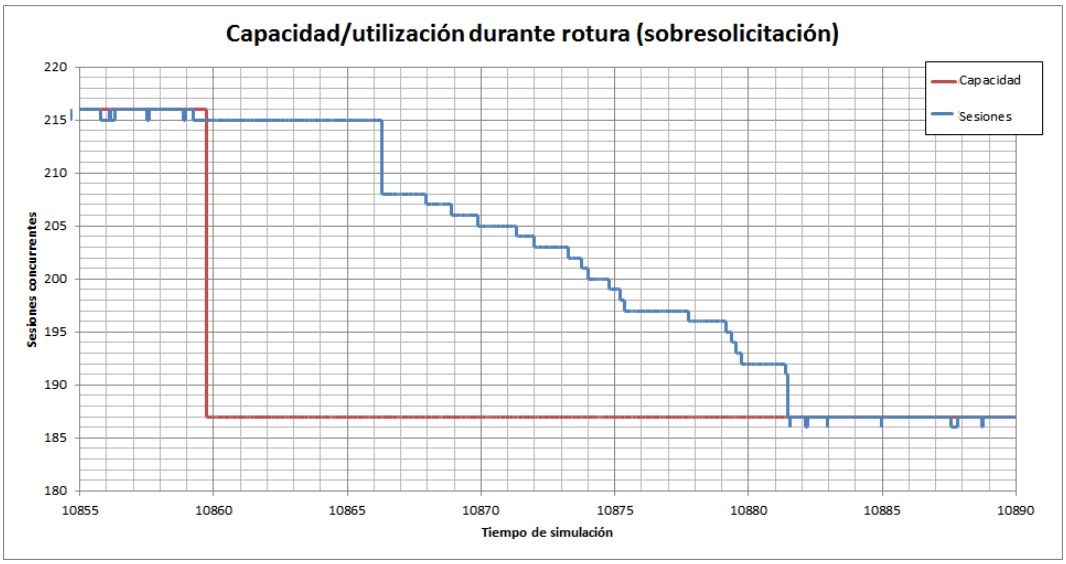

Gráfico 32: Detalle de utilización durante una rotura en sobresolicitación

5.4.6 Comportamiento en condiciones de carga ascendente

En condiciones de carga ascendente existen dos factores que afectan al comportamiento del clúster y que tienen una alta interacción entre sí: la pérdida de capacidad por el fallo de uno o varios de los nodos y el incremento, al mismo tiempo, de la demanda de los clientes al estar en una situación de incremento de carga.

Este es el caso más desfavorable, ya que se produce un efecto contrario al deseado, que sería de incremento de capacidad. Dependiendo del momento del fallo, el clúster puede responder de dos formas diferentes:

a) Si el fallo se produce al principio de la rampa de carga, se producirá un retardo en el incremento de carga admitida mientras se enciende un nuevo nodo, no planificado hasta el momento.
b) Si el fallo se produce al final de la rampa de carga, existe el problema de que la capacidad total se ha reducido. Las sesiones admitidas terminarán, durante un tiempo transitorio de violación del SLA, y después, mediante control de admisión, se regulará la utilización del clúster con relación a la nueva capacidad.

Los experimentos realizados se centran en el segundo caso, más problemático porque incluye el control de admisión para nuevas sesiones. El primer caso está cubierto por el resto de pruebas, en particular en las pruebas de carga intermedia (ver secciones 5.4.3 y 5.4.4).

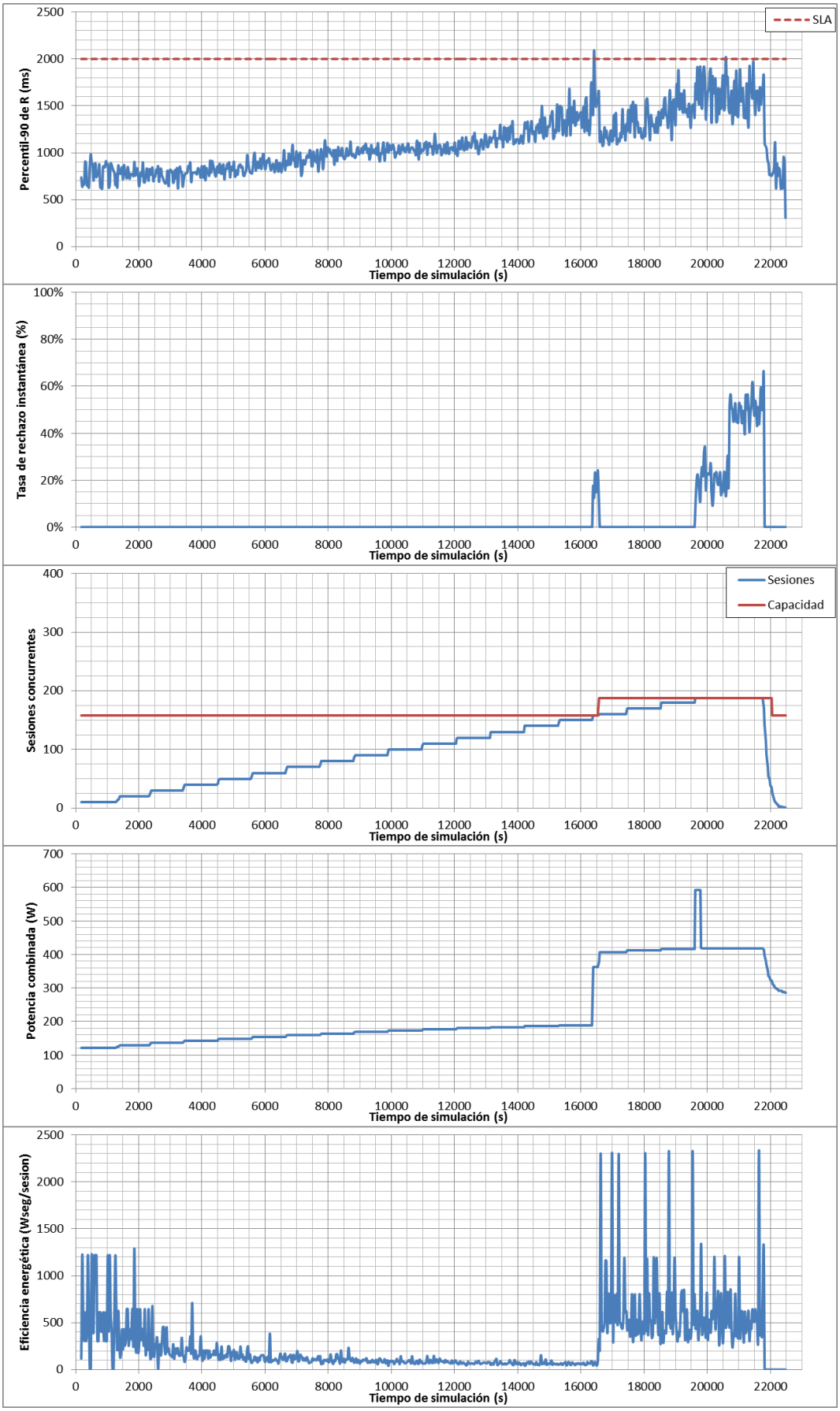

Gráfico 33: Métricas de prueba en incrementos de carga (I)

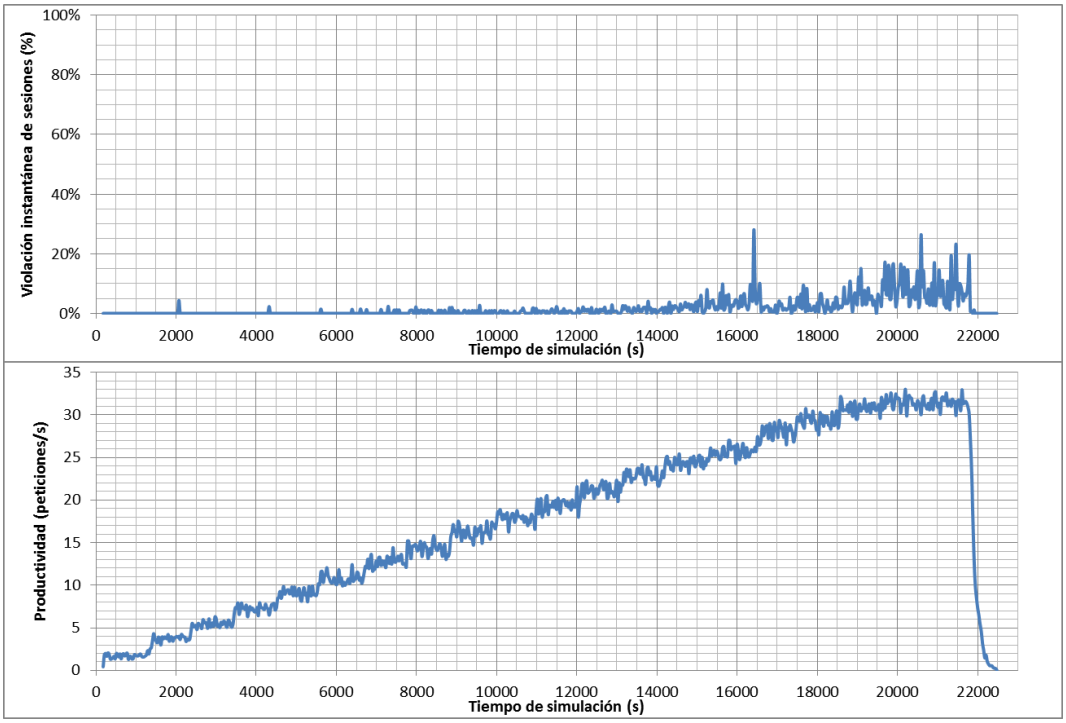

Gráfico 34: Métricas de prueba en incrementos de carga (II)

En esta prueba, cuyos resultados se resumen en el Gráfico 33 y el Gráfico 34, se observa cómo el problema sucede en el segundo 19 000 de ejecución.

En esta prueba, la carga inyectada tiene una forma ascendente lineal que va desde los 1 usuarios hasta los 220. Es decir, cubre todos los regímenes de funcionamiento del clúster. Hacia el final de la rampa de carga se produce el fallo de un nodo.

Repasando los resultados desde el principio de la prueba, se observa cómo la carga empieza a incrementarse y hacia el segundo 17 000 se produce el encendido del segundo nodo del clúster para dar cabida a la nueva carga que se está produciendo.

Continuando con la dinámica del gestor de energía, como la carga sigue incrementándose, en el segundo 19 000 se inicia el encendido de un nuevo nodo. Esta capacidad no está disponible porque el nodo se está arrancando, pero sí que está consumiendo energía.

Durante el arranque de este nodo, antes de que la capacidad se pueda utilizar, el nodo se rompe y nunca llega a formar parte del conjunto de nodos activos del clúster.

En este caso, la detección de la rotura se produce en el gestor de encendido, que nunca llega a detectar la disponibilidad del nuevo nodo y cuando el temporizador de encendido vence, se marca en estado BROKEN.

A partir de ese punto, el consumo del nodo roto no se tiene en cuenta para la optimización del clúster y deja de formar parte de los conjuntos de nodos a manejar. De esta forma, se reduce la capacidad total del clúster, y se ha de aplicar control de admisión para el resto de la prueba, pues la capacidad solicitada es mayor a la disponible.

5.4.7 Comportamiento de condiciones de carga descendente

Las situaciones en las que la carga está decreciendo son especialmente interesantes para el caso en el que el gestor de tolerancia a fallos tiene la posibilidad de rescatar uno de los nodos que está en descarga para suplir el nodo que ha fallado.

Este caso es particularmente favorable, ya que, en principio, la calidad de servicio no se ve afectada, salvo en el caso en que el nodo caído tenga más capacidad que el nodo que lo sustituye, caso en el que podría ser necesario, además del rescate, encender un nodo de los que hay apagados, si éstos están disponibles.

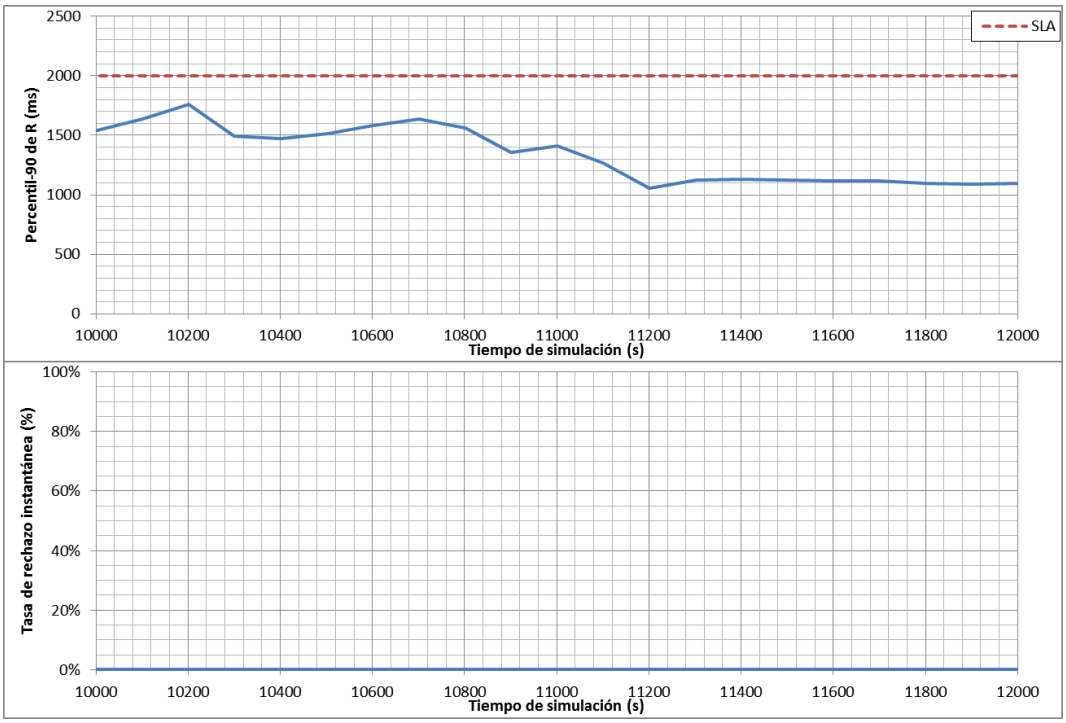

Gráfico 35: Métricas de prueba en carga descendiente (I)

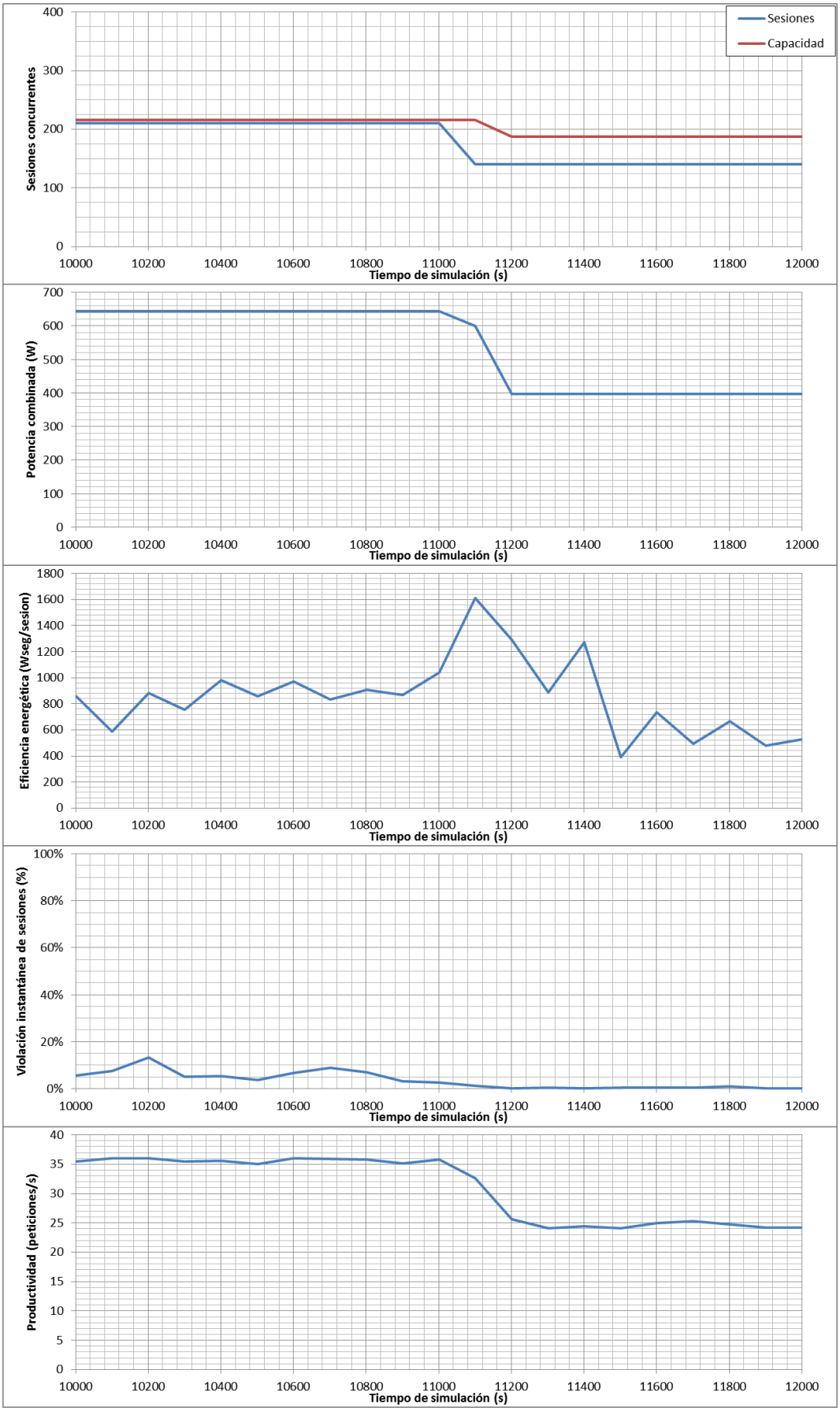

Gráfico 36: Métricas de prueba en carga descendiente (II)

El experimento cuyos resultados se presentan en el Gráfico 35 y el Gráfico 36 se centra en el estudio del fallo de un nodo en una situación de decremento de carga. En este experimento se ha planificado un fallo de un nodo justo después de la orden de apagado de otro nodo, pues esto permite analizar el *rescate* de nodos que se encuentran en descarga.

En concreto, es necesario estudiar el comportamiento del sistema ante dos eventos sucesivos: la reducción de carga que comienza en el segundo 11 000 y la rotura del nodo unos 100 segundos después.

50 segundos después del comienzo de la reducción de carga, se tiene una utilización del clúster tal que permite apagar uno de los nodos cumpliendo las garantías de calidad de servicio y por tanto ahorrar el consumo de uno de los nodos. El gestor energético pone dicho nodo en descarga antes de poder apagarlo. Mientras este nodo está en descarga, otro de los nodos se rompe.

La respuesta del algoritmo es, inmediatamente, sacar al otro nodo de descarga para reponer la capacidad perdida (de forma que la curva de capacidad total no se ve afectada). Poco después, como la carga sigue bajando, se apaga este nodo para terminar con un clúster formado por un solo nodo activo.

5.4.7.1 Análisis del período transitorio

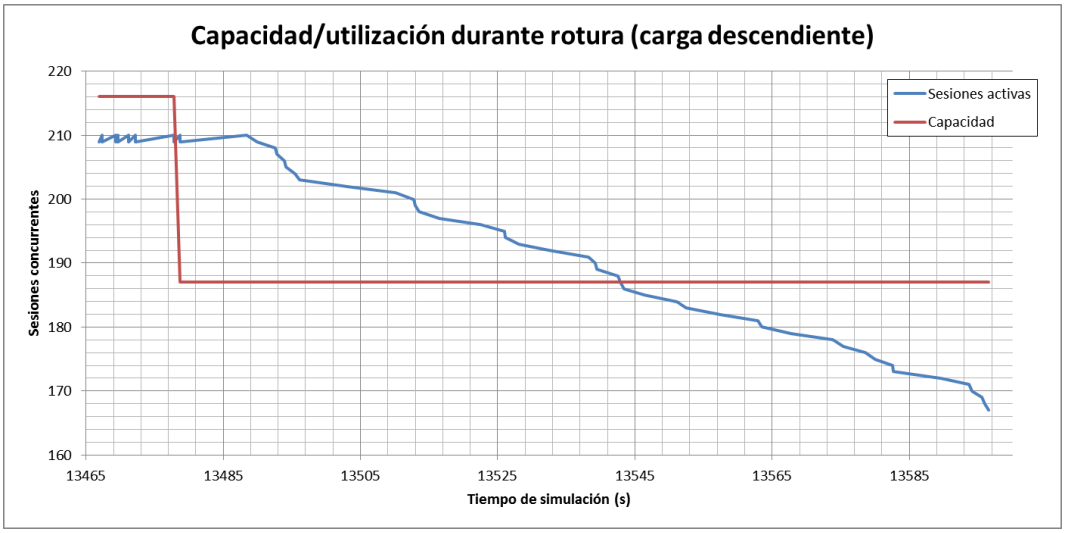

Gráfico 37: Detalle de utilización durante una rotura con carga descendente

En el Gráfico 37 se muestra una rotura de uno de los nodos que coincide con el inicio de un período de reducción de carga. Como se puede observar, existe un período transitorio donde la

cantidad de sesiones en proceso es superior a la cantidad de sesiones admisible. Conforme las sesiones van terminando la carga se reduce hasta la nueva capacidad disponible.

6 CONCLUSIONES

En este trabajo fin de máster se ha estudiado la gestión autonómica de energía y fallos en clústeres de computadores. Un análisis del estado del arte demostró que no existen técnicas que permitan gestionar autonómicamente la energía y los fallos teniendo en cuenta, al mismo tiempo, el cumplimiento de los SLA. Esto motivó la realización de este trabajo con el objetivo de obtener una técnica que cubriese todos estos aspectos.

Se partió de un sistema de gestión energética de clústeres desarrollado en un proyecto fin de carrera anterior. Este sistema tenía dos limitaciones básicas con respecto a la técnica perseguida en este trabajo fin de máster: no trataba correctamente los clústeres heterogéneos y no consideraba la gestión de fallos.

Por lo tanto, en este trabajo se ha presentado, en primer lugar, la ampliación de la gestión energética para que considere correctamente clústeres heterogéneos, lo que se ha logrado modificando el algoritmo anterior en dos aspectos: la selección de qué nodo apagar o encender y el cálculo del umbral de pre-apagado dinámico. En ambos aspectos ahora se tiene en cuenta no sólo la utilización de cada nodo sino también su eficiencia energética. Las mediciones realizadas en este trabajo fin de máster muestran cómo el algoritmo modificado funciona correctamente con clústeres heterogéneos.

Con respecto a la gestión de fallos, el sistema de tolerancia a fallos presentado en este trabajo se integra de forma óptima con el gestor de energía ya desarrollado. Este módulo de gestión de tolerancia a fallos se basa en utilizar las ventajas que proporciona el despliegue de un gestor de energía con provisión dinámica de nodos, utilizando los nodos no activos como nodos de respaldo para la recuperación de los que puedan fallar.

Principalmente, las ventajas de tener un gestor de carga que es capaz de manejar de forma dinámica el conjunto de nodos del clúster es la potencialidad de optimización que presenta

mediante la provisión variable de nodos, permitiendo reducir el consumo energético e incrementando la disponibilidad.

Esto proporciona un mecanismo de tolerancia a fallos que no requiere de instalación de redundancia de forma explícita para la tolerancia a fallos o, expresado de otra forma, permite utilizar la capacidad extraordinaria para, además de tolerancia a fallos, absorber picos de carga en momentos determinados.

La principal aportación al estado del arte de este trabajo es la integración en un único gestor de los tres pilares de la computación autonómica: garantía de calidad de servicio mediante control de admisión, optimización del consumo energético mediante provisión dinámica y maximización de la fiabilidad aprovechando los recursos no utilizados, reduciendo el coste necesario de gestión de redundancia.

Los resultados de las pruebas del gestor de tolerancia a fallos son satisfactorios en el sentido en que muestran cómo el gestor es adaptativo a múltiples situaciones de carga, tanto dinámicas como estacionarias, de forma rápida.

6.1 Conclusiones a las pruebas

Como se puede observar en los diferentes experimentos realizados, el algoritmo de tolerancia a fallos supone una extensión importante del gestor energético que mejora en gran medida la componente autonómica de todo el sistema, siendo capaz de hacer que responda de forma automática ante fallos de los nodos, además de mejorar la calidad de servicio ofrecida a los clientes mediante un servicio más robusto.

El impacto global en términos de violaciones del SLA durante los periodos transitorios es moderado, teniendo un impacto mayor conforme la duración de las sesiones se incrementa.

Adicionalmente, los tiempos transitorios se corresponden con los esperados, de forma que son previsibles, estimables y controlables. Como consecuencia esto permite minimizar las violaciones del SLA a un tiempo transitorio mínimo.

Los objetivos de proporcionar un sistema integrado de gestión autonómica que integre la garantía del SLA, la optimización energética y maximice la tolerancia a fallos se han cumplido y validado, proporcionando un nuevo mecanismo integrado que permite la gestión de los tres componentes en los que basa la gestión autonómica de energía.

6.1.1 Duración del período transitorio

En las pruebas realizadas, ésta recuperación es particularmente rápida, durando aproximadamente la mitad de la longitud de las sesiones:

Longitud de media de sesión	Duración media de sesión	Duración media del período transitorio
10 transacciones	67 s	37 s
25 transacciones	172 s	85 s
100 transacciones	699 s	353 s

Tabla 3: Resumen de duración del tiempo transitorio

Como se puede observar, la duración media de los períodos transitorios es la mitad de la duración de la sesión, cuyo número de transacciones se distribuye de forma exponencial con la media indicada en la Tabla 3.

6.1.2 Impacto en la garantía del SLA

Es complicado establecer una relación entre la garantía del SLA y los diversos casos de pruebas mostrados anteriormente que permita extrapolar resultados a casos intermedios, debido a la no linealidad de los mismos. La Tabla 4 muestra los resultados en términos de la métrica 2 (ver sección 5.3.2) usada para resumir las pruebas:

Situación	Tasa de rechazo	Incumplimientos ($Métrica_2$)
Media carga	0%	0%
Alta utilización	33,52%	0,441%
Sobresolicitación	87,89%	0,435%
Carga ascendente	10,10%	0,539%
Carga descendente	0%	0,446%

Tabla 4: Garantía del SLA en los casos de prueba

Tal y como se observa en las pruebas, sólo se producen rechazos de sesiones si éstas han incluido situaciones de sobresolicitación, siendo la prueba en la que existe sobresolicitación la más desfavorable en términos de rechazo de sesiones.

Si se analizan los resultados en términos de violación del SLA, se observa cómo la tasa total de violación es muy reducida, produciéndose fundamentalmente en situaciones de reconfiguración de la topología del clúster.

El caso más desfavorable, como ya se avanzó en la descripción de la prueba, es la situación de incrementos de carga, donde la tasa de rechazo es 0,1 puntos superior la de las demás pruebas.

El motivo de este resultado es la propia característica de la prueba, mientras se reduce la capacidad, se incrementa la carga, haciendo los procesos de reconfiguración un poco más largos y acentuando la aplicación del control de admisión en los tiempos transitorios.

6.2 Publicaciones

Los primeros resultados de este trabajo han sido publicados en un artículo de congreso, presentado el 20 de julio de 2011 en la conferencia *IADIS International Conference Informatics 2011* organizada en Roma.

Tanto el artículo como el programa de la conferencia pueden encontrarse anexos a este documento. Adicionalmente, se está trabajando en un artículo para revista que engloba todo el último trabajo realizado en el ámbito de la tolerancia a fallos.

6.3 Trabajo futuro

Uno de los aspectos más importantes a realizar es el análisis exhaustivo de la fiabilidad en función de los nodos operativos del clúster y de los nodos disponibles, así como el manejo de más estados energéticos, tales como hibernación y suspensión.

7 REFERENCIAS

[Aversa00] Load Balancing a Cluster of Web Servers Using Distributed Packet Rewriting. Aversa, L. and Bestavros. Computer Science Department, Boston University (2000).

[Aweya02] An adaptive load balancing scheme for web servers. Aweya, J., Ouellette, M., Montuno, D.Y., Doray, B. and Felske, K. International Journal of Network Management, 12. 3—39 (2002)

[Bertini07] Statistical QoS guarantee and energy-efficiency in web server clusters. Bertini, L., Leite, JCB and Mossé, D. 19th Euromicro Conference on Real-Time Systems, 2007. ECRTS'07. 83—92 (2007)

[Bertini10a] Power optimization for dynamic configuration in heterogeneous web server clusters. Bertini, Luciano, Leite, Julius C.B., Mossé, Daniel. 2010. Journal of Systems and Software, 83(4). pp. 585-598. 0164-1212.

[Bertini10b] Power and performance control of soft real-time web server clusters. Bertini, Luciano, Leite, Julius C.B., Mossé, Daniel. 2010. Information Processing Letters. 0020-0190.

[Bevilacqua99] A Dynamic Load Balancing Method on a Heterogeneous Cluster of Workstations. Belvilacqu, A. INFN Bolognia (1999).

[Borthakur07] The Hadop Distributed File System: Architecture and Design. Borthakur, D. The Apache Software Foundation (2007)

[Candellini99] Dynamic load balancing on Web server systems. Candelini, V., Collajani, M. and Yu, P.S. IEEE Internet Computing (1999)

[ChenJ11] Power Management Schemes for Heterogeneous Clusters under Quality of Service Requirements. Chen, J., Huan, K. and Thiele, K. ACM SAC '11 Taiwan (20110029

[ChenY05] Managing server energy and operational costs in hosting centers. Chen, Yiyu, Das, Amitayu, Qin, Wubi, Sivasubramaniam, Anand, Wang, Qian, Gautam, Natarajan. 2005. ACM SIGMETRICS Performance Evaluation Review, 33(1). pp. 303-314. 1595930221.

[Cherkasova02] Session-Based Admission Control: A Mechanism for Peak Load Management of Commercial Web Sites. Cherkasova, L. and Phaal, P. IEEE Transacions on Computers, 51(6). pp. 699-685. (2002)

[Dean08] MapReduce: Simplified Data Processing on Large Clusters. Dean, J. and Ghemaway, S. Communications of the ACM. 51(1) 107-113 (2008)

[Elnikety04] A Method for Transparent Admission Control and Request Scheduling in E-Commerce Web Sites. Elnikety, S., Nahum, E., Tracey, J., Willy, Z. World Wide Web Conference 2004.

[Elnozahy03a] Energy-efficient server clusters. Elnozahy, Mootaz, Kistler, Michael, Rajamony, Ramakrishnan. 2003. Proceedings of Power-Aware Computer Systems. pp. 179-197. Springer.

[Elnozahy03b] Energy conservation policies for web servers. Elnozahy, M., Kistler, M. and Rajamony, R. Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems, 4 (2003)

[Fox97] Cluster-Based Scalable Network Services. Fox, A., Gribble, S.D., Chawathe, Y., Brewer, E.A. and Gauthier, P. ACM SOSP '97. (1997)

[Garcia09] Self-Adjustment Strategy for Models used in Autonomic Transactional Systems. García, Daniel F., Valledor, Pablo, Entrialgo, Joaquín, Medrano, Ramón, et al. Proceedings of the 9th WSEAS International Conference on Applied Informatics and Communications (AIC '09). 2009.

[Garcia10] A self-managing strategy for balancing response time and power consumption in heterogeneous server clusters. Garcia, D.F. and Entrialgo, J. and Garcia, J. and Garcia, M. International Conference On Electronics and Information Engineering (ICEIE), 1 (2010)

[Guerra08] Attaining soft real-time constraint and energy-efficiency in web servers. Guerra, R., Leite, J. and Fohler, G. ACM SAC ’08 (2008)

[Horn01] Autonomic computing: IBM's perspective on the state of information technology. Horn, Paul. IBM Corporation, 2001. 15. pp. 1-39.

[James09] Adaptive Fault Tolerance for Scalable Cluster Computing in Space. James, M.L., Shapiro, A.A., Springer, P.L., Zima, H.P. International Journal of High Performance Computing Applications August 2009 vol. 23 no. 3 227-241

[Kalbarczyk99] Chameleon: A Software Infrastructure for Adaptive Fault Tolerance. Kalbarczyk, Z.T., Iyer, R.K., Bagchi, S. and Whisnant, K. IEEE Transactions on Parallel and Distributed Systems, 10(6). 560—579 (1999)

[Kandaswamy08] Fault tolerance and Recovery of Scientific Workflows on Computational Grids. Kandaswamy, G. and Mandal, A. and Reed, D. IEEE International Symposium on Cluster Computing and Grid (2008).

[Kephart03] The vision of autonomic computing. Kephart, Jeffrey O., Chess, David M., 2003. IEEE Computer, 36(1). pp. 41-50. 0018-9162.

[Kim09] Power-aware Provisioning of Cloud Resources for Real-time Services. Kim, K.H., Beloglazov, A. and Buyya, R. MGC ’09 (2009)

[Kusic09] Power and performance management of virtualized computing environments via lookahead control. Kusic, Dara, Kephart, Jeffrey O., Hanson, James E., Kandasamy, Nagarajan, Jiang, Guofei. 2009. Cluster Computing, 12(1). pp. 1-15. 1386-7857.

[Lefurgy03] Energy management for commercial servers. Lefurgy, C. and Rajamani, K. and Rawson, F. and Felter, W. and Kistler, M. and Keller, T.W. IEEE Computer, 36(12). 39—48 (2003)

[Lu01] Comparing system level power management policies. Lu, Y.H. and De Micheli, G. IEEE Design & Test of Computers, 18(2). 10—19 (2001)

[Medrano11] Gestión energética basada en métricas de calidad de servicio. Medrano, Ramón and García, Daniel and Entrialgo, Joaquín. 2011. Escuela Politécnica de Ingeniería de Gijón. Proyecto 1102015.

[Murphy09] Dynamic Provisioning of Virtual Organization Clusters. Murphy, M.A., Kagey, B., Fenn, M. and Goasguen, M. Clemson University (2009)

[Nagarajan07] Proactive fault tolerance for HPC with Xen virtualization. Nagarajan, Arun Babu and Mueller, Frank and Engelmann, Christian and Scott, Stephen L. Proceedings of the 21st annual international conference on Supercomputing (2007)

[Narendran97] Data Distribution Algorithms for Load Balanced Fault-Tolerant Web Access. Narendran, B., Rangarajan, S. and Yajnik, S. Lucent Technologies (1997)

[Nowka08] A 32-bit PowerPC System-on-a-Chip With Support for Dynamic Voltage Scaling and Dynamic Frequency Scaling. IEEE Journal of Solid-State Circuits, 37(11) (2002)

[Petri95] Load Balancing and Fault Tolerance in Workstation Clusters Migrating Groups of Communicating Processes. Petri, S. and Langerdörfer, H. TU Braunschweig (1995)

[Pinheiro01] Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems. Pinheiro, Eduardo, et al. 2001. 2001. Workshop on Compilers and Operating Systems for Low Power.

[Ramos08] C-Oracle: Predictive Thermal Management for Data Centers. Ramos, L. and Bianchini, R. Rutgers University (2008)

[Ropars11] On the Use of Cluster-Based Partial Message Logging to Improve Fault Tolerance for MPI HPC Applications. Thomas Ropars, Amina Guermouche, Bora Uçar, Esteban Meneses, Laxmikant V. Kalé and Franck Cappello. EURO-PAR 2011 PARALLEL PROCESSING (2011)

[Rusu06] Energy-efficient real-time heterogeneous server clusters. Energy-efficient real-time heterogeneous server clusters. Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium. 418—428 (2006)

[Sharifiana08] A content-based load balancing algorithm with admission control for cluster web servers. Saeed Sharifiana, Seyed A. Motamedia, Mohammad K. Akbarib. Future Generation Computer Systems. Volume 24, Issue 8, October 2008, Pages 775-787 (2008)

[Shye09] PLR: A Software Approach to Transient Fault Tolerance for Multicore Architectures. Shye, A., Blomstedt, J., Moseley, T., Reddi, V., Connors, D. IEEE Transactions on Dependable and Secure Computing (2009).

[Vishnu09] An Effficient Hardware-Software Approach to Network Fault Tolerance with InfiniBand. Vishnu, A., Krishnan, M. and Panda, D.K. CLUSTER '09 (2009)

[Wagle11] Distributed middleware reliability and fault tolerance support in system S. Wagle, Rohit and Andrade, Henrique and Hildrum, Kirsten and Venkatramani, Chitra and Spicer, Michael. Proceedings of the 5th ACM international conference on Distributed event-based system (2011)

[Wang10] Opportunities and Challenges to Unify Workload, Power, and Cooling Management in Data Centers. Wang, Z., Tolia, N. and Bash, C. HP Labs (2010)

[Wang11a] PARTIC: Power-Aware Response Time Control for Virtualized Web Servers. Wang, Y., Wang, X., Chen, M. IEEE Transactions on Parallel and Distributed Systems, 22(2). 2011.

[Wang11b] Coordinating Power Control and Performance Management for Virtualized Server Clusters. Wang, X. and Wang, Y. IEEE Transactions on Parallel and Distributed Systems, 22(2). 2011.

[WangX08] Cluster-level feedback power control for performance optimization. Wang, X. and Chen, M. IEEE 14th International Symposium on High Performance Computer Architecture- 101—110 (2008)

[Weiser94] Scheduling for reduced CPU energy. Weiser, M. and Welch, B. and Demers, A. and Shenker, S. Mobile Computing. 449—471 (114, published 1996)

[Woods07] An Ontology for the Quality of Experience Framework. Woods, J., Siller, M. and Zapopan C.G. IEEE International Conference on Systems, Man and Cybernetics (2007)

[Xue07] An energy-efficient management mechanism for large-scale server clusters. Xue, Z. and Dong, X. and Ma, S. and Fan, S. and Mei, Y. The 2nd IEEE Asia-Pacific Service Computing Conference. 509—516 (2007)

[Yong01] Comparison of Load Balancing Strategies on Cluster-based Web Servers. Yon, T.M. and Ayani, R. Transactions of the Society for Modeling and Simulation, 2001.

[Yum02] Integration of Admission, Congestion and Peak Power Control in QoS-Aware Clusters. Yum, K.H., Jin, Y., Kim, E.J., Das, C.R. IEEE International Conference on Cluster Computing, 2002.

[Yung00] Quantitative Comparison of Power Management Algorithms. Yung, H.L., Chung, E., Simunic, E., Benini, L. and De Micheli, G. Stanford University, 2000.