Universidad de las Ciencias Informáticas

Métricas de rendimiento para el Gestor de Documentos Administrativos eXcriba 2.0 en la Red de Centros de Desarrollo de la Universidad de las Ciencias Informáticas

Trabajo de Diploma para optar por el
Título de INGENIERO EN CIENCIAS INFORMÁTICAS

Autora:
Arianne Valdés Alvarez

Tutores:
Ing. Dayelis Blanco Hernández
Ing. Marcel Sánchez Góngora

La Habana, Junio de 2013

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro de Informatización Universitaria de la Universidad de las Ciencias Informáticas, para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Arianne Valdés Alvarez

_____

Firma del Autor

Dayelis Blanco Hernández

_____

Firma del Tutor

Marcel Sánchez Góngora

_____

Firma del Tutor

Agradecimientos

En este trabajo de diploma he contado con la colaboración de un grupo de compañeros a quienes no puedo obviar. Sería imperdonable.

Siento la necesidad de expresar muestras de gratitud hacia:

Mis tutores Dayelis y Marcel por las horas dedicadas y por su esfuerzo.

Mis padres y mi abuela por todo el apoyo en esta etapa de culminación de los estudios. Por sus sabios consejos y por ser mi sostén al emprender esta tarea.

Oscar por su apoyo incondicional, comprensión, dedicación y ayuda durante los momentos más difíciles.

Michel por las horas dedicadas y sus consejos.

Reinier por su orientación y apoyo.

María Cristina y Bertha Elena por su ayuda.

Mayelín por su apoyo y ayuda.

A ellos reitero el más profundo agradecimiento.

Dedicatoria

A mis queridos padres y a mi súper abuela.

Resumen

El rendimiento como atributo de calidad posibilita caracterizar el comportamiento de un sistema en su entorno de producción. Medir el rendimiento de una aplicación, le permite a las organizaciones tener una percepción de cuántos recursos deben dedicar para que el *software* realice sus funciones óptimamente o también, determinar cuánta carga de trabajo puede procesar el sistema por unidad de tiempo.

Por otra parte, al equipo de desarrollo le permite realizar una valoración crítica sobre los problemas que pudieron afectar la calidad del producto durante su implementación; así como establecer los requerimientos de *hardware* mínimos necesarios que necesita el sistema para llevar a cabo sus funciones.

La investigación tiene como objetivo determinar las métricas de rendimiento del Gestor de Documentos Administrativos eXcriba 2.0 desplegado en la Red de Centros de Desarrollo de la Universidad de las Ciencias Informáticas, mediante la ejecución de pruebas de rendimiento, lo que permitirá la caracterización del desempeño del sistema en su entorno de producción.

En el trabajo se sintetizan los referentes teóricos relacionados con el rendimiento y las pruebas de rendimiento, se definen las métricas que se evaluaron en la aplicación y se realiza un estudio de las herramientas que pueden ser utilizadas para automatizar el proceso de prueba, entre las que se seleccionó JMeter. Además se aplican las actividades definidas para realizar este tipo de prueba, a partir de lo cual se obtuvieron las métricas de rendimiento del GDA eXcriba 2.0 desplegado en la Red de Centros de la Universidad de las Ciencias Informáticas y se comprobó que su desempeño no es el adecuado.

Palabras clave: atributos, desempeño, eXcriba, métricas, pruebas, rendimiento

Índice

Índice de Tablas

Índice de Figuras

Introducción

La calidad es un indicador que incide en todas las esferas de la vida. En la industria del *software* la calidad se puede definir como el grado en el que un sistema, componente o proceso de *software* satisface requerimientos específicos y expectativas o necesidades de clientes y usuarios (1). Desarrollar *software* de calidad que cumplan con los estándares internacionales y las exigencias de los usuarios finales, no es una tarea fácil. En la calidad de un producto informático inciden tanto los requisitos funcionales como los no funcionales. Los primeros, enfocados en la funcionalidad del sistema y los segundos, en las restricciones que debe satisfacer (interfaz, *hardware*, *software*, seguridad, disponibilidad, entre otros).

Los requisitos no funcionales son primordiales para el éxito de un producto de *software*. Los usuarios finales aprecian que un sistema proporcione facilidad de uso y una apariencia vistosa, que esté disponible siempre que se necesite o que ejecute las peticiones en el menor tiempo posible. Estos requisitos no funcionales son definidos por varios autores[1] en términos de requerimientos de atributos de calidad; ineludiblemente, son características que el sistema debe poseer y que contribuyen al logro de la calidad del producto desarrollado.

Cada autor defiende su punto de vista sobre cuáles son los atributos de calidad para un *software*, siendo la lista algo numerosa cuando se agrupan todos. No obstante, quedan a elección del equipo que desarrolla el producto y del cliente, los requisitos que obligatoriamente deberá satisfacer la aplicación. Para un usuario es muy importante que el sistema tarde el menor tiempo posible en proporcionar una respuesta cuando ha ejecutado una acción. Si satisface sus expectativas, percibe que el *software* tiene un buen desempeño; aunque realmente, involucra más que este único factor.

El rendimiento o desempeño (en inglés *performance*), permite caracterizar el comportamiento de un sistema en el entorno de producción[2] y determinar mediante qué restricciones cumplirá con las necesidades de los usuarios, haciendo un uso racional de los recursos disponibles. Es un atributo de calidad que puede incidir negativamente en el funcionamiento de una aplicación, si no se tuvo en cuenta en cada una de las etapas de su ciclo de vida.

Un buen rendimiento en un sistema se manifiesta cuando el *software* es capaz de procesar gran cantidad de peticiones en cortos períodos de tiempo o cuando utiliza eficientemente los recursos disponibles para ello. El rendimiento determina cuán rápido se realizan las operaciones del negocio bajo circunstancias

[1] El Instituto de Ingeniería de Software (por sus siglas en inglés SEI), la Organización Internacional de Normalización (por sus siglas en inglés ISO), Ian Gorton (Miembro *Senior* del Equipo Técnico del Instituto de Ingeniería de Software), Robert Grady, por solo citar algunos.

[2] Se refiere al lugar donde será utilizada la aplicación por el usuario final.

específicas (2). En una época tan agitada como la contemporánea, es común escuchar la frase "el tiempo es oro"; pues las organizaciones demandan aplicaciones que automaticen sus procesos de negocios en el menor tiempo y con el menor costo o esfuerzo posible.

El rendimiento usualmente se descuida hasta que un cliente informa problemas con la aplicación; en otros casos, no se evalúa hasta que se realizan las pruebas o el despliegue inicial del sistema. En cualquiera de las referidas variantes, la solución no es aumentar los requerimientos de *hardware* (3). Por otra parte, resulta costoso arreglar estos problemas puesto que se pierde productividad, mercado y el prestigio de la solución y de la organización; causa insatisfacción en los clientes y provoca aumento en los costos de mantenimiento (4). Por tanto, lo más aconsejable es tener en cuenta el rendimiento desde las primeras etapas del ciclo de vida del *software*.

Los elementos expuestos y el incremento en complejidad de las aplicaciones empresariales que se utilizan hoy por la mayoría de las organizaciones, contribuyen a afirmar que lograr un buen rendimiento en un *software* asegurará un gran por ciento del éxito del producto. Debido a ello, instituciones como el Marco de Desarrollo de la Junta de Andalucía han dedicado grandes esfuerzos en ofrecer asesoría respecto al tema. Se han implementado numerosas soluciones en esta área con el objetivo de medir, evaluar e incluso, proponer soluciones a los problemas de rendimiento que enfrentan hoy un gran número de sistemas.

La Universidad de las Ciencias Informáticas (UCI) es un centro de estudios que, como parte de su misión, se enfoca en producir aplicaciones y brindar servicios informáticos teniendo en cuenta en gran medida la obtención de soluciones que cumplan con los estándares de calidad. Tiene concebido una Red de Centros de Desarrollo que sustenta toda la plataforma productiva de la institución. En el Centro de Informatización Universitaria (CENIA) de la Facultad 1 se ha desarrollado el Gestor de Documentos Administrativos (GDA) eXcriba 2.0; dicho sistema se encuentra desplegado en el centro de datos de la universidad con el propósito de automatizar el proceso de gestión documental para el área de producción de *software*.

El GDA eXcriba 2.0 utiliza los servicios que provee Alfresco[3] como repositorio de contenidos. Presenta una arquitectura en capas que permite la incorporación de nuevas funcionalidades según las necesidades de los clientes. El producto, como parte de las actividades finales de su ciclo de desarrollo, estuvo sometido a un conjunto de pruebas funcionales que permitieron verificar y validar que su funcionamiento fuera el adecuado, sin embargo, no se le realizaron pruebas a nivel de sistema que permitieran conocer con exactitud el rendimiento real de la aplicación.

[3] Alfresco es un Gestor de Contenido Empresarial (*Enterprise Content Management*, en inglés, abreviado ECM) que permite la gestión de todo tipo de documentos. Es una infraestructura multiplataforma de código abierto basada en Java.

A partir de la observación de las trazas del sistema y del propio intercambio con los usuarios, se aprecia que la aplicación no se ha explotado suficientemente en cada uno de los Centros de Desarrollo de la UCI, lo que provoca que aún no se tenga las evidencias necesarias para evaluar el comportamiento de la aplicación en el entorno de producción. A pesar del inconveniente anterior, se han detectado algunos problemas que impiden el correcto funcionamiento del GDA eXcriba 2.0: cuando crece el volumen de información se ralentiza el sistema; así mismo, los tiempos de respuestas a determinadas peticiones aumentan, provocando molestias e insatisfacción en los usuarios.

Por otra parte, el equipo de desarrollo del GDA eXcriba 2.0 desconoce cuántos usuarios conectados simultáneamente puede soportar la aplicación y cuál es la utilización real de los recursos (procesador, red, memoria, disco), tampoco tiene determinados los tiempos de respuesta de las peticiones que se realizan al servidor. Las incidencias descritas anteriormente tributan a que el rendimiento del sistema se vea afectado y que por tanto surja como **problema de investigación:**

Se carece de una concepción definida de las métricas de rendimiento del Gestor de Documentos Administrativos eXcriba 2.0 en la Red de Centros de Desarrollo de la Universidad de las Ciencias Informáticas que permita caracterizar el desempeño del sistema.

Se identifica como **objeto de estudio:** la Ingeniería de Rendimiento de *Software*.

Para proporcionar una solución al problema de investigación se define como **objetivo general:**

Determinar las métricas de rendimiento del Gestor de Documentos Administrativos eXcriba 2.0 desplegado en la Red de Centros de Desarrollo de la Universidad de las Ciencias Informáticas, mediante la ejecución de pruebas de rendimiento que permitan la caracterización del desempeño del sistema.

Se delimita como **campo de acción:** las pruebas de rendimiento en el Gestor de Documentos Administrativos eXcriba 2.0.

Para la consecución del objetivo general se han definido como **objetivos específicos:**

1. Fundamentar los referentes teóricos relacionados con el rendimiento como atributo de calidad y las pruebas de rendimiento.
2. Evaluar el rendimiento del GDA eXcriba 2.0 en la Red de Centros de Desarrollo de la Universidad de las Ciencias Informáticas, teniendo como base los elementos teóricos estudiados acerca de las actividades definidas para realizar pruebas de rendimiento.
3. Definir parámetros de configuración del GDA eXcriba 2.0 en Red de Centros de Desarrollo de la Universidad de las Ciencias Informáticas para mejorar el rendimiento del sistema.

Además, se definen las siguientes **tareas de investigación:**

1. Caracterización del rendimiento como atributo de calidad para esclarecer los conceptos teóricos que permitan su definición.
2. Caracterización de las pruebas de rendimiento como área del conocimiento de la Ingeniería de Rendimiento de *Software* para seleccionar las que se realizarán en la investigación.
3. Análisis de las potencialidades y debilidades de las herramientas para realizar pruebas de rendimiento, de modo que permita seleccionar la más adecuada para ejecutar las pruebas al Gestor de Documentos Administrativos eXcriba 2.0.
4. Identificación del entorno de prueba a partir de la descripción del entorno de producción para simular el proceso de prueba.
5. Definición de los criterios de aceptación de rendimiento del GDA eXcriba 2.0 que permitan establecer los parámetros que deberá satisfacer el sistema durante el proceso de prueba.
6. Diseño de los casos de prueba que permiten describir cada uno de los escenarios que se simularán durante las pruebas.
7. Configuración del entorno de prueba teniendo en cuenta los requerimientos de *hardware* y de *software*, así como las herramientas a utilizar durante el proceso de prueba.
8. Ejecución de pruebas de rendimiento para el GDA eXcriba 2.0 en el entorno de prueba y en el entorno de producción y análisis de los resultados.
9. Ajuste de parámetros de configuración en el GDA eXcriba 2.0 en el entorno de producción para mejorar el rendimiento del sistema
10. Descripción de las métricas de rendimiento del GDA eXcriba 2.0 en la Red de Centros Desarrollo de la Universidad de las Ciencias Informáticas.

Para alcanzar los fines de la investigación se utilizaron los siguientes métodos:

- **Métodos empíricos** (5)**:**
    - Simulación: Método cuantitativo experimental que se utilizó con el propósito de realizar pruebas de rendimiento al Gestor de Documentos Administrativos eXcriba 2.0 en un entorno de prueba con características similares al entorno de producción, lo que permitió obtener parámetros de referencia para el posterior análisis del desempeño del sistema en la Red de Centros de Desarrollo de la Universidad de las Ciencias Informáticas.
    - Caso de estudio: Método cuantitativo observacional mediante el cual se pudo evaluar y monitorear el desempeño del Gestor de Documentos Administrativos eXcriba 2.0 en la Red de

Centros de Desarrollo de la Universidad de las Ciencias Informáticas, lo que permitió obtener las métricas de rendimiento del sistema.

- **Métodos lógicos:**
    - Analítico-Sintético: Permitió extraer la esencia útil a partir de grandes volúmenes de información consultada acerca del rendimiento como atributo de calidad, las pruebas de rendimiento y las herramientas que permiten automatizar el proceso de prueba; lo que favoreció el entendimiento y la claridad de los conceptos implicados en la investigación. Sobre la base del análisis, permitió además, la concreción y objetividad en los diferentes temas expuestos.

Justificación de la investigación:

La evaluación del rendimiento del Gestor de Documentos Administrativos eXcriba 2.0 en la Red de Centros de Desarrollo de la Universidad de las Ciencias Informáticas permitió determinar las métricas de rendimiento y caracterizar el desempeño del sistema en el entorno de producción. Además, la investigación propone una vía para medir el rendimiento del sistema en un contexto determinado, que puede ser adaptado a otros entornos. También ofrece los elementos que se deben tener en consideración para evaluar el rendimiento y las herramientas que se pueden utilizar durante el proceso de prueba.

La investigación se **estructura** en las siguientes secciones:

- Introducción.
- Capítulo I: Fundamentación de los referentes teóricos relacionados con el rendimiento y las pruebas de rendimiento.
- Capítulo II: Descripción de las actividades de las pruebas de rendimiento aplicadas al Gestor de Documentos Administrativos eXcriba 2.0.
- Capítulo III: Métricas de rendimiento para el Gestor de Documentos Administrativos eXcriba 2.0.
- Conclusiones.
- Referencias Bibliográficas.
- Bibliografía.

Capítulo I: Fundamentación teórica

En el presente capítulo se detallan los tópicos sobre los que se sustenta la investigación. Se explican los aspectos fundamentales que definen a la Ingeniería de Rendimiento de *Software* como disciplina y se caracteriza el rendimiento como atributo de calidad. También se esclarecen los elementos teóricos relacionados con las pruebas de rendimiento y las actividades que se deben desarrollar para aplicar las pruebas. Además, se realiza un estudio y comparación de las herramientas que permiten automatizar el proceso de prueba.

1.1 El rendimiento como atributo de calidad

La calidad del *software* se describe comúnmente en términos de atributos de calidad. La IEEE define **atributo de calidad** como un rasgo o característica que afecta la calidad de un elemento. En una jerarquía de atributos de calidad, los de mayor nivel pueden ser llamados factores de calidad; los de menor nivel, atributo de calidad (1). Los atributos de calidad afectan el comportamiento en tiempo real de un sistema, su diseño o la experiencia del usuario. Existen numerosos atributos de calidad, entre los que se pueden mencionar usabilidad, disponibilidad, portabilidad, interoperabilidad, rendimiento, escalabilidad, seguridad, entre otros (6); cada uno de los cuales proporciona una característica distintiva al *software*.

Es importante definir y priorizar cuáles atributos de calidad, ineludiblemente, deberán estar presentes como cualidades o capacidades de la aplicación, pues su impacto difiere en cada sistema. Además, la mejor manera de garantizar que la aplicación exhiba dichas características es implementando acciones en cada una de las etapas del ciclo de vida del *software*; para ello es imprescindible que se hayan definido los atributos de calidad desde que se concibe la idea de desarrollar el sistema. Al final, los atributos de calidad deben ser cuantificables en sus especificaciones mediante escalas de medida prácticas y apropiadas (7).

El rendimiento como atributo de calidad posibilita caracterizar el comportamiento de un sistema en su entorno de producción. Medir el rendimiento de una aplicación, le permite a las organizaciones tener una percepción de cuántos recursos deben dedicar para que el *software* realice sus funciones óptimamente o también, determinar cuánta carga de trabajo puede procesar el sistema por unidad de tiempo. Por otra parte, al equipo de desarrollo le permite realizar una valoración crítica sobre los problemas que pudieron afectar la calidad del producto durante su implementación; así como establecer los requerimientos de *hardware* mínimos necesarios que necesita el sistema para llevar a cabo sus funciones.

El estudio de bibliografías de referencia en esta área del conocimiento, permite identificar que no existe una taxonomía única que defina al rendimiento como atributo de calidad; por lo que queda a consideración del investigador cuál es la corriente por la cual se inclinará.

1.1.1 Enfoques de rendimiento

La IEEE define **rendimiento** como el grado en que un sistema o componente realiza sus funciones teniendo en cuenta determinadas restricciones como velocidad, precisión o uso de memoria (1).

Len Bass, Paul Clements y Rick Kazman afirman que el **rendimiento** define la capacidad de respuesta del sistema, es decir, el tiempo necesario para responder a estímulos (eventos) o el número de eventos procesados en un cierto intervalo de tiempo. Además, exponen que las cualidades de rendimiento se expresan a menudo mediante el número de transacciones por unidad de tiempo que procesa el sistema o por la cantidad de tiempo que se tarda en completar una transacción (8).

En FURPS+ (9), acrónimo utilizado para referirse a la clasificación de atributos de calidad propuesta por Robert Grady, se define que el **rendimiento** impone condiciones a los requisitos funcionales, tales como la velocidad, la eficiencia; mide el tiempo de respuesta, la capacidad de procesamiento, el consumo de los recursos, el rendimiento efectivo total y la eficacia.

En la norma ISO 9126-1 (10) se utiliza la terminología **eficiencia (*efficiency*)**, la cual se refiere a la cantidad de recursos utilizados por el sistema al proporcionar una funcionalidad requerida. Ejemplo: la cantidad de espacio en disco, memoria o ancho de banda utilizado ante una petición. Engloba los siguientes sub-atributos:

- Comportamiento en el tiempo (*time behavior*): Caracteriza los tiempos de respuesta para una capacidad de procesamiento determinada del sistema. Ejemplo: la tasa de transacción.
- Comportamiento de recursos (*resource behavior*): Caracteriza la cantidad de recursos utilizados y la duración de su uso en la ejecución de su función.

Ian Gorton (2) define el **rendimiento** como la medida que indica la cantidad de trabajo que una aplicación debe realizar en un tiempo determinado, y/o el plazo de tiempo que debe cumplirse para ejecutar una operación correcta. Engloba los siguientes sub-atributos:

- Capacidad de procesamiento (*throughput*): Es la medida de la cantidad de trabajo que una aplicación debe realizar por unidad de tiempo. Usualmente se mide en transacciones por segundos (tps) o mensajes procesados por segundos (mps). Medir solamente la capaidad de procesamiento promedio de una aplicación sería un grave error, se debe tener en consideración también el máximo valor de capacidad de procesamiento que puede soportar el sistema y prepararlo para soportar este límite. Por ejemplo, una banca en línea debe garantizar poder ejecutar alrededor de 1 000 tps de los clientes de Internet; un sistema de gestión de inventario de un gran almacén puede que tenga que procesar alrededor de 50 mps de los asociados que solicitan órdenes diariamente.

- Tiempo de respuesta (*response time*): Es la medida de la latencia[4] de una aplicación al procesar una petición. No obstante, se debe definir el tiempo límite que podría demorar una respuesta y el promedio de tiempos de respuesta entre los que pudiera oscilar para atender cada petición. Por ejemplo, el 95% de todas las peticiones deben ser procesadas en menos de 4s y ninguna deben tardar más de 15s.
- Tiempo límite (*deadlines*): Define el tiempo máximo que puede tardar un sistema para ejecutar determinadas funcionalidades.

El Instituto de Ingeniería de Software (11) (por sus siglas en inglés SEI) define que el **rendimiento** caracteriza la eficacia del servicio que provee el sistema. Se clasifican dentro de este atributo como requerimientos de rendimiento:

- Latencia (*latency*): Intervalo de tiempo que transcurre desde que se produce un evento hasta que se ejecuta una respuesta del sistema. Este intervalo está dado por un tiempo de inicio (latencia mínima) y un tiempo final (latencia máxima).
- Capacidad de procesamiento (*throughput*): Define el número de respuestas de eventos que deben ser completadas totalmente en un intervalo de tiempo especificado.
- Capacidad (*capacity*): Es una medida de la cantidad de trabajo máximo que un sistema puede realizar. En términos de redes se le conoce como ancho de banda, lo cual usualmente se mide en megabits por segundos (Mbps). Una definición más práctica puede ser la máxima capacidad de procesamiento que se puede lograr sin violar los requerimientos de latencia especificados (12).
- Modo (*mode*): Puede ser caracterizado como el estado de la demanda del sistema y el estado del sistema: la configuración de los recursos utilizados para satisfacer la demanda.

Teniendo en cuenta cada una de las taxonomías propuestas anteriormente para el rendimiento, se decide proponer la definición formal de rendimiento sobre la cual se acoge la presente investigación y cuáles métricas serán evaluadas para caracterizar el desempeño del Gestor de Documentos Administrativos eXcriba 2.0:

Rendimiento: Define el grado en que un sistema o componente realiza sus funciones teniendo en cuenta determinadas restricciones como la capacidad de procesamiento, el tiempo de respuesta y la utilización de los recursos:

[4] Tiempo que transcurre entre un estímulo y la respuesta que produce, siendo un estímulo una petición al sistema y la respuesta, el servicio que provee el software para satisfacerla.

- Capacidad de procesamiento: Define el número de peticiones que pueden ser procesadas totalmente en un intervalo de tiempo especificado o la cantidad de información procesada por unidad de tiempo. Usualmente se mide en transacciones por segundos (tps) o mensajes procesados por segundos (mps), aunque puede medirse también en términos de (Mb/s).
- Tiempo de respuesta o latencia: Intervalo de tiempo que transcurre desde que se realiza una petición hasta que se ejecuta una respuesta del sistema. Se mide en segundos (s).
- Utilización de recursos: Caracteriza la cantidad de recursos utilizados y la duración de su uso en la ejecución de su función. Los recursos pueden ser memoria, procesador, disco y red.

1.1.2 Relación entre métricas de rendimiento

Según Roger Pressman, la conjugación de algunas métricas de rendimiento, permite definir un conjunto de condiciones de prueba dentro de los límites operativos normales de un sistema:

- N: Número de usuarios concurrentes.
- T: Número de transacciones por usuario por unidad de tiempo (tps).
- D: Cantidad de información procesada por el servidor por transacción (Kb).
- P: La cantidad de información global procesada por unidad de tiempo (Mb/s).

Según se cambia cada condición de prueba se pueden obtener las demás métricas como: la respuesta de usuario promedio, el tiempo promedio para descargar una unidad de datos estandarizada o el tiempo promedio para procesar una transacción. Se determinan a partir de la siguiente ecuación:

$$P = N \times T \times D \quad (13)$$

Para un mejor entendimiento, se explica un ejemplo práctico:

En un momento determinado 20 000 usuarios concurrentes realizan una solicitud (transacción, T) cada dos minutos como promedio a un sitio. Cada solicitud requiere que la aplicación descargue un artículo que promedia 3 KB de longitud. De ahí que la cantidad de información procesada en unidad de tiempo se puede calcular como:

$$\mathrm{P} = 20\,000 \times \frac{1}{120\,s} \times 3\,KB$$

$$\mathrm{P} = 500\,KB/s$$

$$\mathrm{P} = 4\,\mathrm{Mb}/s$$

Luego, se puede concluir que la conexión de red para el servidor tendría que soportar esta tasa de datos.

Se coincide con Brad Micklea[5] en que las métricas tiempo de respuesta (R), capacidad de procesamiento (X) y utilización de los recursos (U) no pueden verse como fenómenos aislados debido a la intrínseca relación entre ellas. A menudo el tiempo de respuesta y la capacidad de procesamiento no están en proporción pues por lo general se pretende maximizar la capacidad de procesamiento cuando el tiempo de respuesta está por debajo o se mantiene sobre cierto umbral. Por ejemplo se quiere aumentar la capacidad de procesamiento del sistema manteniendo el tiempo de respuesta igual o menor a 2 segundos. También se puede observar mediante la figura 1 que la utilización de los recursos puede llegar a controlar el comportamiento del sistema. Cuando existe una saturación de los recursos puede generarse una caída dramática de la capacidad de procesamiento y un aumento proporcional del tiempo de respuesta en la zona crítica. La zona crítica evidencia una severa caída del rendimiento de las aplicaciones, debido a que el sistema emplea la mayor parte de su tiempo en administrar el conflicto de los recursos en lugar de atender las peticiones (14).

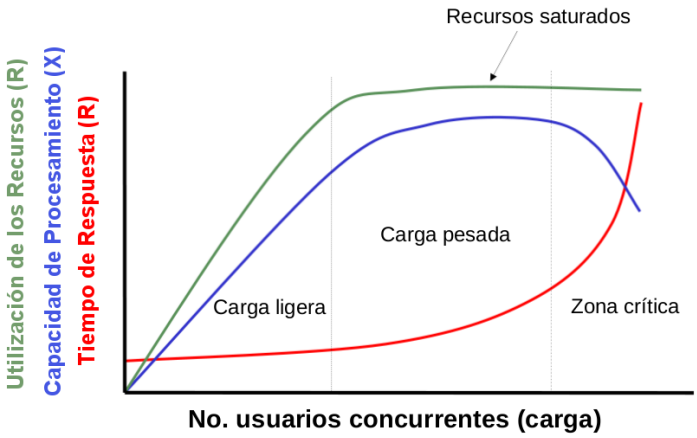

Figura 1: Comportamiento de las métricas básicas del rendimiento (14).

1.2 Ingeniería de Rendimiento de Software

La Ingeniería de Rendimiento de Software (*Software Performance Engineering*, en inglés, abreviado SPE) es el área dentro de la Ingeniería de Software que involucra un conjunto de roles, habilidades, actividades, prácticas, herramientas y resultados aplicados a cada una de las fases del ciclo de vida de un *software* para asegurar que la solución sea diseñada, implementada y soportada operacionalmente para satisfacer los requerimientos no funcionales de rendimiento definidos. El cumplimiento de esos atributos de calidad se valida mediante la supervisión de los sistemas en producción (15).

[5] Brad Micklea es Gerente de Producto de la empresa *Quest Software*. Ha ayudado a las compañías a lograr mejor rendimiento en aplicaciones Java y J2EE por más de 3 años.

En “*The Future of Performance Software Engineering”*, los autores definen que existen dos enfoques asociados a esta disciplina. Un enfoque basado puramente en la medición: en la aplicación de pruebas, diagnósticos y la afinación del *software* a finales de su ciclo de vida, cuando puede ser ejecutado y medido. El otro enfoque se basa en un modelo, donde se crean modelos de rendimiento en las fases tempranas del ciclo de vida del *software* y se utilizan los resultados cuantitativos obtenidos del modelo para ajustar la arquitectura y el diseño, con el propósito de satisfacer los requerimientos de rendimiento (16).

En la presente investigación se sigue el primer enfoque explicado anteriormente debido a que mediante las pruebas de rendimiento se mide el desempeño del GDA eXcriba 2.0 en su entorno de producción; el sistema, desplegado en la Red de Centros de Desarrollo de la UCI, transita por la etapa final de su ciclo de vida. Independientemente de la posición adoptada, a partir del estudio realizado, se defiende la idea que el segundo enfoque es la vía más acertada para desarrollar aplicaciones que exhiban un buen rendimiento en su entorno de producción, asegurando bajos costes de mantenimiento durante el proceso de desarrollo de *software*.

1.2.1 Objetivos de la Ingeniería de Rendimiento de Software

La Ingeniería de Rendimiento de Software proporciona la información necesaria para construir aplicaciones que satisfagan los requisitos de rendimiento de manera adecuada y que se encuentren dentro de lo presupuestado. Se definen como objetivos principales de esta disciplina:

- Aumentar los ingresos comerciales al garantizar que el sistema puede procesar las transacciones dentro de los plazos necesarios.
- Eliminar la falla del sistema que requiere de la cancelación de las actividades de su desarrollo debido a falta objetiva de rendimiento.
- Eliminar la implementación del sistema con retraso debido a problemas de rendimiento.
- Eliminar el reproceso del sistema disponible debido a problemas de rendimiento.
- Eliminar esfuerzos evitables del sistema de afinación.
- Evitar costos adicionales e innecesarios en adquisición de hardware.
- Reducir el incremento de los costos de mantenimiento del software, debido a los problemas de rendimiento en la producción.
- Reducir gastos generales operativos adicionales para el manejo de problemas en el sistema debido a los problemas de rendimiento (15).

1.2.2 Actividades de la Ingeniería de Rendimiento de Software

Cualquier proceso de la Ingeniería de Rendimiento de Software incluye algunas o todas las siguientes actividades:

1 Identificar las preocupaciones. Se identifican los factores que afectan los objetivos de rendimiento.

2 Definición y análisis de requerimientos. Se describe de forma general los requisitos de rendimiento y los escenarios que describen su comportamiento.

3 Predicción del rendimiento de los escenarios, la arquitectura y el diseño mediante el modelado de la interacción del comportamiento con los recursos.

4 Pruebas de rendimiento de una parte o de todo el sistema sobre cargas normales y cargas límites de uso.

5 Mantenimiento y evolución. Se predice el efecto que tendrá en el sistema cambios en su funcionamiento o la adición de nuevas funcionalidades.

6 Análisis total del sistema. Se considera el *software* planificado en el sistema completo y finalmente desplegado (16).

1.2.3 Elementos principales en la Ingeniería de Rendimiento de Software

La Ingeniería de Rendimiento de Software engloba determinados elementos que constituyen la esencia de la disciplina. Los objetivos de rendimiento permiten determinar cuándo una aplicación satisface las metas para su rendimiento. El modelado del rendimiento proporciona un enfoque estructurado y sistemático para satisfacer esos objetivos. Establecer directrices en el diseño y la arquitectura permite realizar ingeniería de rendimiento desde etapas tempranas en el ciclo de vida del *software*. Definir un marco de rendimiento y escalabilidad contribuye a organizar y priorizar los elementos asociados al rendimiento. La medición permite identificar cuánto falta por satisfacer los objetivos identificados. Definir los miembros del equipo que se necesita contribuye a delimitar las habilidades y tareas de cada uno de los roles involucrados. Lo que se ha enunciado se puede sintetizar en la siguiente figura:

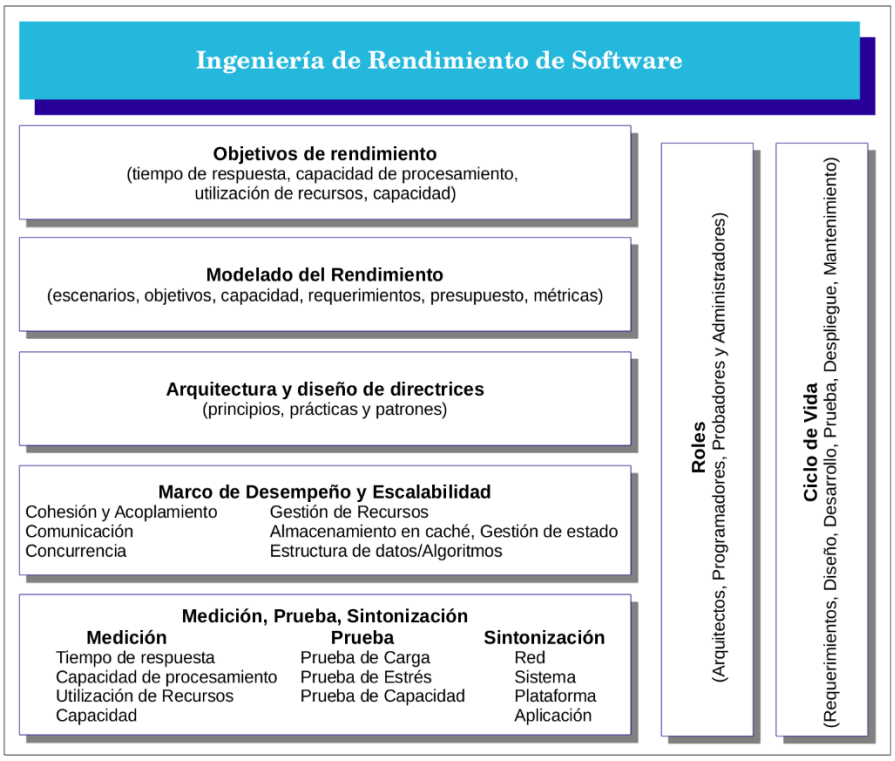

Figura 2: Ingeniería de Rendimiento de Software (3).

1.2.3.1 Definición de los objetivos de rendimiento

Para poder aplicar ingeniería al rendimiento es necesario tener una cultura de rendimiento durante todas las fases del ciclo de vida del *software*. Se debe tener definido cuál es el procedimiento a seguir para lograr cumplir con los objetivos del rendimiento del sistema. Medir cada uno de estos objetivos en cada una de las etapas permite evaluar el avance hacia ellos o cuánto nos alejamos de ellos. Las respuestas a las siguientes preguntas se corresponden con algunos de los objetivos de rendimiento que se puden definir para una aplicación:

- ¿Con qué rapidez se necesita que funcione un sistema?
- ¿En qué punto el rendimiento del sistema se vuelve inaceptable?
- ¿Cuál es el por ciento en procesamiento o memoria utilizado por el sistema?

Determinar estos objetivos permite crear una línea base para el rendimiento de una aplicación. En resumen pueden ser: el tiempo de respuesta o latencia, la capacidad de procesamiento y la utilización de recursos.

1.2.4 Áreas de la Ingeniería de Rendimiento de Software

La Ingeniería de Rendimiento de Software abarca cinco áreas del conocimiento según las define Walter Kukets en "*Enterprise Software Performance Engineering"* (10):

- El Ciclo de Vida del Desarrollo de Software (*Software Development Lifecycle*, en inglés, abreviado SDLC) y la Arquitectura (por sus siglas en inglés SA). Esta área proyecta el rendimiento en cada una de las etapas del ciclo de vida del *software* (Definición, Diseño, Construcción, Prueba y Despliegue), como vía para lograr obtener un buen rendimiento del producto construido.
- Validación y Pruebas de Rendimiento (*Performance Validation and Test*, en inglés, abreviado PVT). Ésta área se inserta en la fase de pruebas del ciclo de vida del *software* con el propósito de medir y validar los objetivos del rendimiento definidos para el sistema.
- Planeación de la Capacidad (*Plannig Capacity*, en inglés, abreviado CP). Ésta área se encarga de planear la capacidad de cada uno de los componentes del sistema, de tal manera que se determina la capacidad en cuanto a recursos que utiliza el sistema, se analiza la capacidad real requerida y se planifica un futuro uso de los recursos.
- Gestión de Rendimiento de Aplicación (*Application Performance Management*, en inglés, abreviado APM). Se encarga de monitorear el rendimiento de aplicaciones complejas distribuidas teniendo en cuenta la experiencia de usuarios finales.
- Detección y Resolución de Problemas (*Problem Detection and Resolution*, en inglés, abreviado PDR). Se encarga de detectar cuáles son los problemas que afectan el rendimiento de aplicaciones y proponer soluciones, tanto en un ambiente pre-producción[6] como en un entorno de producción u operacional del *software*.

La investigación se centra en el área Validación y Pruebas de Rendimiento dentro la Ingeniería de Rendimiento de Software y se enmarca en la actividad definida Pruebas de Rendimiento con el propósito de medir el desempeño del sistema en su entorno de producción. Aclarar que el proceso no se realizará como se ha aconsejado, durante la etapa de pruebas al *software*, debido a que el sistema se encuentra en la etapa de mantenimiento.

1.3 Las pruebas de rendimiento dentro del área de pruebas de software

El ciclo de vida de un *software* consta de varias etapas, cada una de las cuales garantizan un por ciento

[6] Se refiere a la etapa del ciclo de vida del *software* cuando todavía no ha sido desplegada la solución en el ambiente de producción (el entorno de trabajo real donde se utilizará por los usuarios finales).

de la calidad del producto final. El equipo de desarrollo es el actor principal de este proceso y el encargado de concebir, diseñar, implementar, probar, desplegar y mantener la solución. Una etapa esencial, que se encarga fundamentalmente de detectar los errores que se pudieron cometer en el ambiente pre-producción, es la prueba de *software*.

1.3.1 Pruebas de software

"Las pruebas de *software* son un elemento crítico para la garantía de la calidad del *software* y representan una revisión final de las especificaciones, del diseño y de la codificación" (17). Hetzel define la prueba de *software* como cualquier actividad dirigida a evaluar un atributo o capacidad de un programa o sistema y determinar que cumple con los resultados que se requieren (18). Por otra parte, Myers plantea que las pruebas de *software* constituyen el proceso de ejecutar un programa o sistema con la intención de encontrar errores (19).

Luego, se puede inferir que las pruebas de *software* constituyen el conjunto de actividades o procesos llevados a cabo con el propósito de evaluar tanto los requerimientos funcionales como los no funcionales de un programa o sistema para detectar fallas, errores o defectos. Por tanto, la realización de las pruebas es la manera más adecuada para determinar el grado de calidad de un producto de *software*.

1.3.1.1 Objetivos de las pruebas de software

"Una prueba de *software* no puede asegurar la ausencia de defectos (…)" (20), de ahí que persigan los siguientes objetivos:

- Descubrir un error en el sistema o errores no detectados hasta entonces.
- Construir buenos casos de pruebas: con una alta probabilidad de mostrar errores no encontrados hasta el momento.
- Demostrar hasta qué punto las funcionalidades del *software* parecen funcionar de acuerdo con las especificaciones y alcanzar los requisitos propuestos.
- Verificar y validar los factores de calidad de *software*[7].

1.3.1.2 Estrategias de pruebas de software

Resulta conveniente considerar el proceso de Ingeniería de *Software* como se muestra en la siguiente imagen:

[7] Según Hetzel los factores pueden ser exactitud, fiabilidad, usabilidad, integridad, eficiencia, capacidad de ser probado, documentación, estructura, flexibilidad, reusabilidad y capacidad de ser mantenible (18).

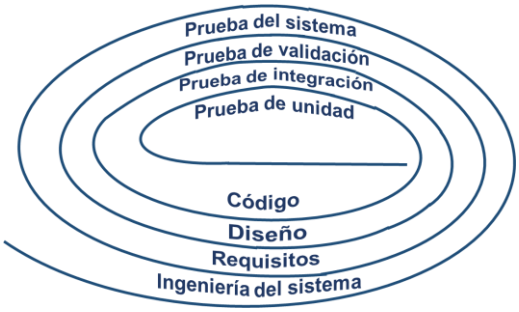

Figura 3: Estrategias de prueba (21).

Se coincide con Roger Pressman en que al principio, la ingeniería del sistema define el papel del *software* y lleva el análisis de los requisitos de este, donde se establecen el dominio de información, la función, el comportamiento, el desempeño, las restricciones y los criterios de validación del *software*. Al desplazarse hacia el interior de la espiral se llega al diseño y, por último, a la codificación. También es posible ver una estrategia para la prueba del *software* pero desde adentro hasta el exterior de la espiral; a continuación se caracteriza cada uno de los momentos de la estrategia (21):

1. **Pruebas de unidad:** Se centran en la verificación de la menor unidad del diseño del *software*: el componente de *software* o módulo; en el procesamiento lógico interno y en las estructuras de datos dentro de las interfaces de un componente; se pueden realizar en paralelo para varios componentes.

2. **Pruebas de integración:** Se centran en el diseño y la construcción de la arquitectura del *software*. Persiguen como objetivo seleccionar los módulos probados mediante pruebas de unidad y construir una estructura de programa que esté de acuerdo con lo que establece el diseño. Se pueden aplicar como enfoques para realizar las pruebas la integración descendente o ascendente. Además se pueden realizar:

    - Las pruebas de regresión consisten en ejecutar un conjunto de pruebas que se han llevado a cabo anteriormente para asegurar que los cambios no han propagado efectos colaterales no deseados.

    - Las pruebas de humo se realizan de manera continua con el propósito de descubrir errores que impiden a la “construcción” (los componentes que han sido implementados e integrados) realizar su funcionamiento adecuadamente.

3. **Pruebas de validación:** Se centran en la detección de errores pero enfocados a los requisitos que son necesarios para el usuario final. Comienzan tras culminar las pruebas de integración, cuando se han ejercitado los componentes individuales, se ha ensamblado el *software* como paquete, se han detectado y corregido errores de interfaz. Estas pruebas pueden ser de tipo alfa

(cuando se lleva a cabo por un cliente en el entorno de desarrollo) o beta (cuando se lleva a cabo por los usuarios finales del *software* en los lugares de trabajo de los clientes).

4. **Pruebas de sistema:** Se centran en ejercitar profundamente el software, verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones adecuadas. Se distinguen como tipos de prueba:

    - Las pruebas de recuperación fuerzan el fallo del *software* de muchas maneras y verifican que se recupera apropiadamente.
    - Las pruebas de seguridad se realizan para verificar que los mecanismos de protección incorporados al sistema responden correctamente.
    - Las pruebas de rendimiento se diseñan para probar el rendimiento de un *software* en tiempo de ejecución dentro de contexto de un sistema integrado. En el siguiente epígrafe se detallará en qué consiste y los tipos de prueba que involucra.

1.3.2 Prueba de rendimiento

Las pruebas de rendimiento (en inglés *Performance Testing*) constituyen un área dentro de la disciplina de Ingeniería de Rendimiento de Software. Permiten identificar cuellos de botella[8] en las aplicaciones, establecer una línea base para poder realizar futuras pruebas y determinar si se cumplen los objetivos de rendimiento. Además, contribuyen a recolectar toda la información que le puede ser útil al cliente para comprobar la calidad del producto luego que fue probado. Los resultados obtenidos de las pruebas y el análisis pueden ayudar a estimar las configuraciones del hardware que soportará el sistema una vez desplegado en el entorno real de trabajo (20).

Existen diversas definiciones acerca de pruebas de rendimiento, por ejemplo:

Definición 1: Investigación técnica para determinar o validar las características de velocidad, escalabilidad y/o estabilidad de un producto bajo prueba (20).

Definición 2: Prueba conducida para evaluar la conformidad de un sistema o componente con los requerimientos de rendimiento especificados (1).

Definición 3: Estudio del comportamiento de un sistema cuando se ve sometido a una carga que actúa de manera concurrente.

[8] Puntos donde se producen congestión y retrasos en una aplicación, ralentizando la tramitación de las solicitudes y causando que los usuarios experimenten retrasos inaceptables en el servicio. Puntos débiles de una aplicación. Obstáculos.

- Sistema: Objeto sobre el que se prueba. Puede ser una aplicación, un componente, una plataforma de hardware.
- Carga: Indicador sobre la concurrencia del sistema.
- Concurrencia: Puede hacer referencia a:
    - Las acciones funcionales realizadas por período de tiempo: consultas por segundo, altas por hora, compras por día, entre otras.
    - Procesos ejecutados de manera simultánea: ejecuciones de un programa, accesos a bases de datos, entre otros.
    - Usuarios validados en una aplicación en un momento determinado (22).

Teniendo en cuenta las anteriores definiciones, se puede sintetizar que una prueba de rendimiento es el conjunto de actividades o procesos llevados a cabo para evaluar y medir los objetivos de rendimiento especificados para un *software* o un componente de él. Las pruebas de rendimiento se ejecutan tanto para determinar cómo responde un sistema ante una cierta carga, como para validar otros atributos relacionados con la calidad, como pueden ser la escalabilidad o el uso de recursos (23).

1.3.2.1 Objetivos de las pruebas de rendimiento

Algunas de las razones específicas para realizar pruebas de rendimiento pueden ser:

- Predecir o estimar los atributos de rendimiento de una aplicación en un ambiente de producción y evaluar si los problemas de rendimiento expuestos pueden ser o no resueltos, atendiendo a las predicciones.
- Evaluar si la capacidad de la aplicación es suficiente.
- Determinar los recursos que requerirá en un futuro una aplicación para que su rendimiento sea aceptable.
- Determinar las configuraciones idóneas para que el sistema pueda obtener el mejor rendimiento posible.
- Verificar que los requerimientos de rendimiento de la aplicación son realmente los deseados, teniendo en cuenta las limitaciones en la utilización de los recursos.
- Comparar los requerimientos de rendimiento reales de la aplicación con los deseados para verificar el cumplimiento de los objetivos de rendimiento.
- Analizar el comportamiento de la aplicación ante diferentes niveles de carga.

- Identificar cuellos de botella.
- Obtener información relacionada con la velocidad, escalabilidad y estabilidad de la aplicación (20).

1.3.2.2 Actividades para realizar pruebas de rendimiento

La presente investigación utilizará las actividades definidas en “*Performance Testing Guidance for Web Applications*” (24) como procedimiento para llevar a cabo las pruebas de rendimiento en el sistema eXcriba 2.0; debido a la amplia información, organización, estructura y calidad de los contenidos que se presentan. A pesar que en el Capítulo 7 se definen 9 actividades por el modelo de mejora de procesos (por sus siglas en inglés CMMI), se decide que se aplicarán las siguientes actividades principales ya que engloban a groso modo las que desglosa CMMI, sin dejar de tener en cuenta los aspectos que son fundamentales para ejecutar las pruebas:

1. **Identificar el entorno de prueba:** Identificar el entorno físico de prueba y de producción, así como las herramientas y los recursos que utilizará el equipo de prueba. El entorno físico incluye el *hardware*, el *software* y las configuraciones de red.
2. **Identificar los criterios de aceptación de rendimiento:** Identificar los objetivos de rendimiento que deberá satisfacer la aplicación (dígase tiempo de respuesta, capacidad de procesamiento y utilización de recursos), así como las criterios de aceptación para los objetivos de rendimiento.
3. **Planificar y diseñar las pruebas:** Se debe identificar los casos de pruebas críticos y los principales escenarios; definir los datos de prueba; determinar la ruta de navegación de los escenarios seleccionados y caracterizar la carga de trabajo en el ambiente de producción.
4. **Configurar el entorno de prueba:** Preparar el entorno de prueba, configurar las herramientas y los recursos necesarios para ejecutar la estrategia. Asegurar el ambiente de prueba para instrumentar un monitoreo de recursos si es necesario. Si se utilizan herramientas para automatizar las pruebas es necesario en este punto configurarlas.
5. **Implementar el diseño de las pruebas:** Implementar las pruebas de rendimiento, de acuerdo con el diseño previsto, en la herramienta que se utilizará para automatizar el proceso. Verificar que el entorno de prueba coincide con la configuración y el diseño de prueba y que la simulación de la carga refleja con precisión el diseño de prueba.
6. **Ejecutar las pruebas:** Ejecutar, monitorear y validar los *script* de prueba, así como el sistema y los datos de prueba. Generar los resultados de las pruebas.
7. **Analizar los resultados, elaborar el informe y volver a probar:** Comparar los resultados y los criterios de aceptación. Analizar la información tanto de forma individual, como en un equipo

multifuncional. Cambiar la prioridad de las pruebas restantes y volver a probar si es necesario.

La guía que se ha escogido como referencia explica cómo aplicar las pruebas de rendimiento teniendo en cuenta las características del modelo de mejora de procesos para el desarrollo del *software*, factor importante para la UCI que comienza a dar sus primeros pasos en esta dirección. Por otra parte, su contenido se sustenta en lecciones aprendidas y experiencias obtenidas por expertos en el tema, lo que contribuye a que exista un conjunto de buenas prácticas definidas para realizar los procesos de pruebas de rendimiento de la mejor manera posible.

1.3.2.3 Tipos de pruebas de rendimiento

Existen varios tipos de pruebas de rendimiento para medir el comportamiento de uno o varios objetivos de rendimiento. Realizar estos tipos de pruebas permite mejorar las capacidades de una aplicación y maximizar el beneficio de un negocio. A continuación se relacionan:

- **Pruebas de carga (Load Testing):** Se realizan para verificar el comportamiento de un sistema bajo condiciones de una carga determinada; la cual puede ser el número de usuarios en producción o un número de transacciones durante un tiempo determinado. Permite medir el tiempo de respuesta, la capacidad de procesamiento y los niveles de utilización de los recursos; así como identificar cuellos de botella en una aplicación, asumiendo que se producen bajo condiciones de carga específicas (23).
    - Pruebas de resistencia (Endurance testing): Son un subconjunto de pruebas de carga. Se realizan con el propósito de determinar si la aplicación puede mantener la carga esperada de manera continua y durante un largo tiempo. El objetivo principal de este tipo de prueba es verificar que no existen fugas de memoria o procesos que pierdan rendimiento tras un cierto período de tiempo (23).
- **Pruebas de estrés (Stress testing):** Se realizan con el objetivo de revelar los errores de la aplicación bajo condiciones de carga máxima; mediante la ejecución de un número de usuarios muy superior al esperado o bien por la substracción de recursos. Tiene como finalidad determinar la robustez de una aplicación cuando la carga es extrema, así como el límite real de la aplicación en cuanto a número de usuarios concurrentes, el número de transacciones por segundo, entre otros. En este tipo de pruebas los tiempos de respuesta de la aplicación no son importantes y tienden a ser ignorados (23).
    - Pruebas pico (Spike testing): Se realizan insertando la carga en el sistema en forma de “picos” que se irán lanzando en distintos momentos de la prueba y que permiten comprender el comportamiento de la aplicación ante cambios bruscos de carga (23).

- **Pruebas de capacidad (Capacity testing):** Se realizan para determinar cuántos usuarios y/o transacciones soportará un sistema en un futuro sin dejar de cumplir con los objetivos de rendimiento definidos. Son útiles para planear un futuro crecimiento de usuarios o de información en la base de datos; para poder satisfacer esa demanda, se necesita conocer cuál sería la utilización adicional de los recursos necesarios para soportar futuros niveles de utilización (25).

Se decide realizar las pruebas de carga y estrés en la presente investigación para medir el rendimiento del GDA eXcriba 2.0, poniendo en práctica para ello cada una de las actividades descritas anteriormente.

1.4 Estudio de herramientas para realizar pruebas de rendimiento

Actualmente, la utilización de herramientas se ha hecho imprescindible para realizar disímiles procesos. Las pruebas son un flujo muy importante dentro del proceso de desarrollo de *software* y en ocasiones se requiere utilizar determinadas herramientas para mejorar la calidad y eficiencia de dicho flujo. Existe una gran variedad de soluciones informáticas enfocadas a medir uno o varios objetivos de rendimiento en un sistema, en general, concebidas para ejecutar pruebas de rendimiento. En la presente investigación, para realizar la selección de las herramientas a utilizar con este propósito, se tuvo en cuenta el estudio “Evaluación de herramientas *Free/Open Source* para pruebas de *software*” (26). A continuación, se desglosa el proceso en fases:

1.4.1 Fase 1: Descripción general

Se elige un subconjunto del total de herramientas existentes, que cumplan con un foco específico, y se completan los siguientes datos para cada una de ellas: nombre, descripción y Localizador de Recursos Uniforme (*Uniform Resource Locator*, en inglés, abreviado URL). Las herramientas seleccionadas para el estudio cumplen con la característica de ser libres o de código abierto:

Pylot: Es una herramienta de código abierto para realizar pruebas de rendimiento y escalabilidad de servicios web. Ejecuta pruebas de carga mediante el Protocolo de Transferencia de Hipertexto (*Hypertext Transfer Protocol*, en inglés, abreviado HTTP), lo cual es muy útil para planear la capacidad, realizar una evaluación comparativa y análisis del sistema. Genera concurrencia de solicitudes, verifica las respuestas del servidor y produce reportes teniendo en cuenta métricas. Las pruebas pueden ser ejecutadas y monitoreadas desde una interfaz gráfica o por consola (27).

JMeter: Es una aplicación de escritorio desarrollada en Java, diseñada para medir el rendimiento y el comportamiento de los sistemas ante las pruebas de sobrecarga. Se puede utilizar para probar el rendimiento tanto de los recursos estáticos como dinámicos (archivos, *Servlets*, *scripts* de Perl, *Java Objects*, bases de datos y consultas, servidores FTP, entre otros). También, para simular una sobrecarga en un servidor, una red o un objeto, para poner a prueba su resistencia o para analizar el rendimiento global

para diferentes tipos de carga. Además, permite hacer un análisis gráfico de rendimiento o probar el servidor/*script*/comportamiento del objeto con sobrecargas concurrentes (28).

OpenSTA: Es un conjunto de herramientas para ejecutar pruebas de sobrecarga y medir el rendimiento de aplicaciones sobre plataformas Windows mediante los protocolos HTTP y el Protocolo Seguro de Transferencia de Hipertexto (*Hypertext Transfer Protocol Secure*, en inglés, abreviado HTTPS). Los resultados y estadísticas generados durante las pruebas se pueden recopilar mediante una variedad de mecanismos automatizados o controlados por el usuario. Una vez completadas las pruebas, las trazas pueden ser revisadas, graficadas, filtradas y exportadas para que puedan ser utilizadas por un sistema de generación de reportes más sofisticado (29).

Grinder: Es una aplicación de escritorio con interfaz gráfica diseñada para realizar pruebas de carga y medir el rendimiento. Utiliza el lenguaje de programación Jython (implementación de Python mediante Java) para definir las pruebas. Permite realizar pruebas de carga a cualquier aplicación que tenga una Interfaz de Programación de Aplicaciones (*Application Programming Interface*, en inglés, abreviado API) de Java; utilizando para ello servidores web HTTP, servicios web mediante el Protocolo de Acceso a Objeto Simple (*Simple Object Access Protocol*, en inglés, abreviado SOAP), Transferencia de Estado Representacional (Representational State Transfer, en inglés, abreviado REST) y servidores de aplicaciones como CORBA (en inglés *Common Object Request Broker Architecture*), RMI (en inglés *Remote Method Invocation*), JMS (en inglés *Java Message Service*) y EJB (en inglés *Enterprise JavaBeans*) (30).

1.4.2 Fase 2: Descripción detallada por criterios

Se completan las siguientes tablas de acuerdo con los criterios que se consideran relevantes para realizar la selección, teniendo en cuenta la información proporcionada por el sitio oficial de la herramienta u otros sitios reconocidos:

Tabla 1: Datos generales

Herramienta	Versión	Inicio del Proyecto	Licencia	Plataforma	Interfaz	Lenguaje
Pylot	1.26 – Julio 2009	Octubre 2007	-	Windows, GNU-Linux/Unix	GUI	Python
JMeter	2.9 – Febrero 2013	Diciembre 1998	Apache License (Versión 2.0) es compatible con GPL.	Independiente	GUI	Java
OpenSTA	1.4.4 – Octubre 2007	Febrero 2001	GNU GPL	Windows	GUI	C++

Grinder	3.11 – Octubre 2012	Diciembre 2000	Grinder License	Windows, Linux/Unix	GUI	Jython y Clojure

Tabla 2: Criterios de documentación

Herramienta	Guía de instalación	Manual de usuario	Preguntas frecuentes	Soporte en línea			
				Foro	Lista de correo	Blog	Wiki
Pylot	Sí	Sí	No	No	No	No	No
JMeter	No requiere	Sí	Sí	Sí	Sí	Sí	Sí
OpenSTA	Sí	Sí	Sí	No	Sí	No	Sí
Grinder	No requiere	Sí	Sí	No	Sí	Si	Sí

Tabla 3: Criterios de madurez

Herramienta	Madurez del proyecto	Grado de actualización	Actividad en lanzamientos	Actividad en el reporte de errores
Pylot	R	R	M	M
JMeter	MB	MB	B	MB
OpenSTA	MB	M	M	R
Grinder	MB	B	B	B

1.4.3 Fase 3: Selección

Del conjunto de herramientas propuestas en el estudio se descarta Pylot pues es un proyecto relativamente joven cuyo grado de actualización del producto no es el mejor, así como no se encontraron evidencias actualizadas del reporte de errores y su solución; incluso, la última versión del producto dista bastante de la fecha actual. OpenSTA, a diferencia de Pylot, es un proyecto de varios años de experiencia; no obstante, no presenta un buen nivel de actualización y por tanto la actividad en el lanzamiento de nuevas versiones del producto es casi nula; a lo anterior se le suma que solo es posible utilizarlo en estaciones con Windows, lo cual no es compatible con el ambiente donde se encuentra desplegada la aplicación a probar.

Grinder, a pesar de ser un fuerte candidato, es un proyecto que utiliza Jython y opcionalmente Clojure como lenguajes de programación y para poder configurar las pruebas se requiere de conocimientos de programación en dichos lenguajes. Además, solo soporta HTTP como protocolo para realizar las pruebas; los archivos de propiedades que definen la cantidad de carga a aplicar en las pruebas deben ser configurados manualmente en cada una de las estaciones agentes y si se simula descargas de archivos de gran tamaño repetidamente en varios hilos, pueden ocurrir problemas con la memoria en la estación de trabajo.

Teniendo en cuenta el estudio realizado se decide utilizar JMeter en su versión 2.9. La herramienta se ejecuta independiente a la plataforma de hardware utilizada, ya que es un proyecto 100% Java. Cuenta con un manual de usuario disponible que detalla cada una de sus funcionalidades, con frecuencia se publican nuevas versiones y tiene un buen grado de retroalimentación pues los errores reportados son resueltos en cortos períodos de tiempo. El grado de actualización es excelente con respecto a la fecha en la que se ha realizado el estudio (marzo de 2013) y la curva de aprendizaje es baja, no se requiere de habilidades como desarrollador o conocimientos avanzados para ejecutar las pruebas.

1.4.4 Potencialidades funcionales de JMeter

El análisis de las potencialidades funcionales de la herramienta permitió corroborar que con JMeter es posible simular, con el nivel de concurrencia que se desee, las interacciones de una o varias comunidades de usuarios con un sistema; así como grabar mediante el navegador web las interacciones de un usuario con la interfaz de una aplicación y convertir la grabación en un plan de prueba. Además, la aplicación muestra los resultados de las pruebas en una amplia variedad de informes y gráficas, lo que permite hacer un análisis crítico del rendimiento del sistema y facilita la rápida detección de los cuellos de botella existentes debido a tiempos de respuesta excesivos. JMeter proporciona gráficas ilustrativas sobre el consumo de los recursos en el servidor donde se encuentra alojada la aplicación mientras se realizan las pruebas y permite ejecutar peticiones a un sistema mediante los protocolos: HTTP, HTTPS, el Protocolo Ligero de Acceso a Directorios (*Lightweight Directory Access Protocol,* en inglés, abreviado LDAP), SOAP, la Conectividad de Base de Datos de Java (*Java Database Connectivity*, en inglés, abreviado JDBC), el Protocolo de Oficina Postal (*Post Office Protocol*, en inglés, abreviado POP3) y el Protocolo de Acceso de Mensaje de Internet (*Internet Message Access Protocol*, en inglés, abreviado IMAP); así como medir el rendimiento de aplicaciones en Java mediante JMS.

Los principales componentes de JMeter que se pueden utilizar para implementar un plan de prueba se pueden consultar en el **Anexo 1: Tipos de componentes del plan de prueba en JMeter**. Por otra parte, se ha reportado que con cada instancia simple de JMeter se pueden generar cargas exitosas hasta de 1000 usuarios[9], dependiendo de los recursos de *hardware* que se disponga (27). Finalmente, la herramienta permite realizar pruebas distribuidas y remotas para medir el rendimiento del sistema. Estas pruebas consisten en instalar instancias de JMeter en varios ordenadores como “esclavos”, los cuales adquieren las instrucciones de una instancia de un ordenador con un JMeter con interfaz de usuario funcionado como “maestro” y envían peticiones al servidor web que será probado (28). La siguiente imagen ilustra el proceso:

[9] En JMeter cada hilo se simula mediante hilos (en inglés *threads*).

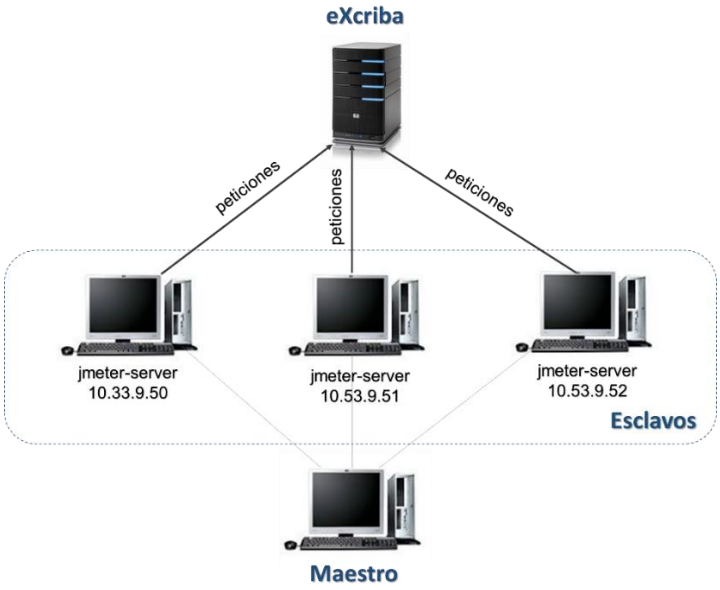

Figura 4: Pruebas remotas y distribuidas con JMeter (28).

1.5 Estudio de herramientas para monitorear la utilización de los recursos durante las pruebas

Para monitorear el comportamiento de una estación de trabajo existen numerosas soluciones que permiten obtener resultados ilustrativos y de manera rápida; lo que facilita el análisis y comprensión de las métricas que caracterizan al *hardware* que se monitorea. Del gran número de herramientas que existen se selecciona un subconjunto; para lo cual se tuvo en cuenta que fueran libres o de código abierto, así como las de mayor aceptación a partir de la búsqueda realizada en Internet:

Nagios: Es una herramienta de código abierto bajo la Licencia Pública General (*General Public License*, en inglés, abreviado GPL) que permite monitorear los componentes de *hardware* (carga del procesador, uso de los discos, trazas del sistema) y de *software* que se especifiquen en una red como los servicios de red a través de los protocolos Protocolo Simple de Transferencia de Correo (*Simple Mail Transfer Protocol*, en inglés, abreviado SMTP), POP3, HTTP, Protocolo para la Transferencia de Noticias en Red (*Network News Transport Protocol*, en inglés, abreviado NTTP), Protocolo de Mensajes de Control de Internet (*Internet Control Message Protocol*, en inglés, abreviado ICMP), Protocolo Simple de Administración de Red (*Simple Network Management Protocol,* en inglés, abreviado SNMP). Facilita la generación de alertas cuando los parámetros definidos por el administrador exceden los márgenes establecidos. Además, permite el monitoreo remoto a través de los túneles cifrados de la Capa de Conexión Segura (*Secure Sockets Layer*, en inglés, abreviado SSL) o SSH (en inglés *Secure SHell*) y en tiempo real a través de una interfaz web, con la posibilidad de generar informes y gráficas del comportamiento de los sistemas; incluso, la visualización de las notificaciones enviadas, el historial de problemas y los archivos de registros. Tiene

soporte para plataformas Windows, GNU-Linux/Unix; la última versión publicada es la 3.5.0 en Marzo 2013. El proyecto ha alcanzado un buen nivel de madurez, frecuentemente publican nuevas versiones y se evidencia en la búsqueda realizada, gran actividad en el reporte y solución de errores. Se facilita la retroalimentación con la comunidad de desarrollo a través de las listas de correo y blogs dedicados para ello, además se proporciona una *wiki* para el acceso a toda la documentación disponible con posibilidades de consultar la sección de Preguntas Frecuentes *(Frequently Asked Questions*, en inglés, abreviado FAQ). URL: http://www.nagios.org/

JavaMelody: Herramienta de código abierto bajo la Licencia Pública General Reducida (*Lesser General Public License*, en inglés, abreviado LGPL) para monitorear servidores de aplicaciones Java o Java EE (en inglés *Enterprise Edition*) en entornos de producción o de Aseguramiento de la Calidad (*Quality Assurance*, en inglés, abreviado QA). Permite medir y calcular estadísticas en un ambiente real de una aplicación en dependencia del uso de la aplicación por los usuarios. Se basa principalmente en las estadísticas de peticiones y en gráficos evolutivos que miden los tiempos de ejecución, el por ciento de errores de peticiones HTTP, Lenguaje de Consulta Estructurado (*Structured Query Language*, en inglés, abreviado SQL) y páginas JSP (en inglés *JavaServer Pages*); el número de usuarios por sesiones, el número de conexiones JDBC, la utilización de la memoria de Java y de la Unidad Central de Procesamiento (*Central Processing Unit*, en inglés, abreviado CPU). Tiene soporte para plataformas Windows, GNU-Linux/Unix; la última versión publicada es la 1.44.0 en Marzo 2013. El proyecto es relativamente joven, solo 4 años de experiencia, pero frecuentemente publica nuevas versiones, aunque en la búsqueda realizada no se evidencia gran actividad en el reporte de errores. Se facilita la retroalimentación con la comunidad de desarrollo a través de las listas de correo, además se proporciona una *wiki* para el acceso a toda la documentación disponible. URL: http://code.google.com/p/javamelody/

JConsole: Herramienta de monitoreo compatible con las Extensiones de Gestión de Java (*Java Management eXtensions*, en inglés, abreviado JMX) bajo la licencia GPL. Utiliza la instrumentación de JMX de la Máquina Virtual de Java (por sus siglas en inglés JVM) para monitorear el rendimiento y el consumo de recursos de las aplicaciones que se ejecutan en la plataforma de Java. Permite realizar el monitoreo de forma local o remota y genera gráficos teniendo en cuenta el consumo de memoria de la JVM, de CPU, los hilos y las clases que son cargadas. Los parámetros pueden ser consultados y configurados a través de una *Shell* o de la interfaz gráfica que proporciona. Para facilitar la retroalimentación con la comunidad de desarrollo se pueden utilizar las listas de correo y los blogs disponibles para ello. URL: http://openjdk.java.net/tools/svc/jconsole/

Munin: Herramienta para el monitoreo de los recursos de red, útil para analizar su comportamiento. Proporciona gran variedad de gráficos a través de una interfaz web; de forma general, permite monitorear

servidores, una Red de Área de Almacenamiento[10] (*Storage Area Network*, en inglés, abreviado SAN) y aplicaciones. Munin tiene una arquitectura maestro-nodo mediante la cual el maestro se conecta a los nodos en intervalos regulares y les solicita información; a continuación almacena los datos en ficheros RDD y si es necesario, actualiza los gráficos. Está programada en Perl; tiene soporte para las plataformas Windows, GNU-Linux, FreeBSD, NetBSD; la última versión publicada es la 2.0.12 en Marzo 2013. Frecuentemente se publican nuevas versiones y se evidencia en la búsqueda realizada, gran actividad en el reporte y solución de errores. Posee una guía de instalación y un manual de usuario disponibles. Se facilita la retroalimentación con la comunidad de desarrollo a través de las listas de correo, además se proporciona una *wiki* para el acceso a toda la documentación disponible con posibilidades de consultar la sección de Preguntas Frecuentes. URL: http://munin-monitoring.org/

El estudio permitió identificar las potencialidades de cada una de las herramientas descritas, por eso se decide proponer para la monitorización de los recursos en el entorno de producción JavaMelody y Nagios. La primera propuesta permite monitorear el comportamiento durante un tiempo prolongado de la Máquina Virtual de Java en el servidor donde se encuentra instalado el GDA eXcriba 2.0 y la segunda, estudia el comportamiento de los recursos de *hardware* y de *software* que se necesite monitorear.

Conclusiones parciales

- La revisión teórica relacionada con el rendimiento como atributo de calidad contribuyó a identificar que no existe una taxonomía única que permita su definición a partir de un mismo precepto. Esta situación generó que la autora se inclinara por definir al rendimiento como el grado en que un sistema o componente realiza sus funciones teniendo en cuenta determinadas restricciones como la capacidad de procesamiento, el tiempo de respuesta y la utilización de los recursos.
- El análisis de las áreas que abarca la Ingeniería de Rendimiento permitió identificar las pruebas de rendimiento como mecanismo para medir el desempeño de las aplicaciones y mediante las cuales es posible determinar las métricas de rendimiento en un sistema.
- El estudio de las actividades que se requieren realizar para ejecutar las pruebas de rendimiento facilitó la comprensión de los elementos que se involucran en el proceso y delimitó el procedimiento que se utiliza en la investigación para evaluar el rendimiento del Gestor de Documentos Administrativos eXcriba 2.0.

[10] Es una red de dispositivos de almacenamiento compartido, tales como arreglos de almacenamiento en disco y sistemas de automatización de cintas. La arquitectura de una SAN permite que los recursos de almacenamiento estén disponibles para varios servidores en una red de área local o amplia (49).

- El análisis de las herramientas para realizar pruebas de rendimiento y el monitoreo de aplicaciones, permitió seleccionar a JMeter para automatizar el proceso de prueba; además proponer a JavaMelody y Nagios para monitorear el comportamiento de la Máquina Virtual de Java y de los recursos de *hardware* y *software* respectivamente, en el entorno de producción.

Capítulo II: Actividades de las pruebas de rendimiento

Teniendo como referencia el estudio realizado en el capítulo anterior, se procede a ejecutar las pruebas de rendimiento de tipo carga y estrés al Gestor de Documentos Administrativos eXcriba 2.0, a partir de las actividades que se definieron en el primer capítulo de la investigación para llevar a cabo estos tipos de pruebas, utilizando la herramienta JMeter en su versión 2.9. En los siguientes epígrafes se detallan cómo se decide realizar cada una de estas actividades. Primero se propone definir los escenarios principales a probar; con JMeter se graban dichos escenarios, y posteriormente, la herramienta genera los *script* que constituyen la base para observar, analizar y determinar las métricas de rendimiento del sistema.
Por otra parte, se ha identificado que es necesario ejecutar tres grupos de pruebas fundamentales para el desarrollo de la investigación. Un primer grupo realizado en el laboratorio de prueba para establecer las medidas básicas del rendimiento del sistema; un segundo grupo ejecutado directamente en los Centros de Desarrollo identificados y un tercer grupo, en los Centros nuevamente pero con los parámetros de configuración de la Máquina Virtual de Java ajustados.

2.2 Identificar el entorno de prueba

El entorno de prueba es el conjunto de *software* y *hardware* mediante el cual el equipo de prueba realiza las pruebas al sistema implementado (29); para su configuración, es necesario definir primero el entorno de producción sobre el que será instalada la aplicación.

2.2.1 Caracterización del entorno de producción

La Universidad de las Ciencias Informáticas tiene concebido para la producción de *software* una Red de Centros de Desarrollo, cada uno de los cuales utiliza el GDA eXcriba 2.0 para automatizar los procesos de gestión documental que tienen lugar en esta área. Para cada área se ha desplegado una instancia del sistema en el centro de datos de la universidad; para ello se han destinado 5 servidores donde se encuentra virtualizado el sistema, un servidor adicional donde se encuentran virtualizadas las bases de datos y una SAN para el almacenamiento. Cada uno de los servidores donde está desplegado el sistema tiene configurado 128 GB de Memoria de Acceso Aleatorio (*Random-Access Memory,* en inglés, abreviado RAM) y 4 *Quad-Core*[11] de CPU. Suman un total de 15 Centros de Desarrollo con parámetros de configuración muy parecidos, de ellos se seleccionaron tres para realizar las pruebas de rendimiento teniendo en cuenta

[11] Un *Quad-Core* es un microprocesador con 4 núcleos, completos e independientes, montados en un único chip.

la frecuencia de uso y los reportes de incidencia recibidos: CISED[12], CDAE[13] y CEGEL[14].

2.2.2 Configuraciones de *hardware* en el entorno de producción

Tabla 4: Configuraciones de *hardware* en los servidores de los Centros de Desarrollo.

Recurso	CISED	CDAE	CEGEL
Sistema Operativo	Ubuntu 10.4		
RAM de la Virtualización	2.0 GB	2.0 GB	8.0 GB
CPU de la Virtualización	Intel(R) Xeon (R) 4 núcleos 2.4 GHz		Intel(R) Xeon (R) 6 núcleos 2.4 GHz
Arquitectura	64 bits		
Ancho de banda	1 Gb/s		
Disco	SAN HP EVA[15] 6100 con ancho de banda de 8 Gb/s, 4 puertos con velocidad de 4 Gb/s, 4 GB de caché y soporta hasta 154 000 peticiones de Entrada/Salida por segundo.		

Tabla 5: Configuraciones de *hardware* del servidor que contiene las bases de datos.

Recurso	CISED	CDAE	CEGEL
Sistema Operativo	Ubuntu 10.4		
RAM de la Virtualización	2.0 GB	2.0 GB	8.0 GB
CPU	Intel(R) Xeon (R) 4 núcleos 2.4 GHz		Intel(R) Xeon (R) 6 núcleos 2.4 GHz
Arquitectura	64 bits		
Ancho de banda	1 Gb/s		
Disco	SAN HP EVA 6100 con ancho de banda de 8 Gb/s, 4 puertos con velocidad de 4 Gb/s, 4 GB de caché y soporta hasta 154 000 peticiones de Entrada/Salida por segundo.		

2.2.3 Configuraciones de *software* en el entorno de producción

En cada servidor donde está instalado eXcriba 2.0 se ejecutan las aplicaciones que se enuncian a continuación:

- Servidor Web Apache.
- Servidor Web Apache Tomcat.

[12] Centro de Identificación y Seguridad Digital.

[13] Centro de Consultoría y Desarrollo de Arquitecturas Empresariales.

[14] Centro de Gobierno Electrónico.

[15] Arreglo Virtual Empresarial, en inglés *Enterprise Virtual Array*, es un sistema de arreglo de almacenamiento de tipo empresarial para englobar y automatizar las tareas de gestión de arreglos de modo que se gestione más capacidad de almacenamiento con menos recursos.

- Servidor Samba.
- Servidor de control de versiones.
- Servidor SSH.
- Solución de salva y restaura de la red: Bácula.
- Herramienta para monitorear los recursos en red: Munin.
- Herramienta para monitorear servidores de aplicación Java o JEE: JavaMelody.
- Máquina Virtual de Java Sun-6.0.

Las configuraciones de la JVM para cada uno de los Centros de Desarrollo son las siguientes:

CISED:

```
JAVA_OPTS='-XX:MaxPermSize=512m -Xms128m -Xmx768m -server -Xcomp
   -Xbatch -Xss1m -XX:+UseConcMarkSweepGC -XX:+CMSIncrementalMode
   -XX:CMSInitiatingOccupancyFraction=80'
```

CDAE:

```
JAVA_OPTS="-XX:MaxPermSize=512m -Xms128m -Xmx768m -server -Xcomp
   -Xbatch -Xss1m -XX:+UseConcMarkSweepGC -XX:+CMSIncrementalMode
   -XX:CMSInitiatingOccupancyFraction=80"
```

CEGEL:

```
JAVA_OPTS='-Xms512m -Xmx2048m -XX:MaxPermSize=512m -server'
```

La descripción de cada uno de los parámetros especificados se puede consultar en el **Anexo 2: Descripción de las opciones de la Máquina Virtual de Java.**

2.2.4 Configuraciones de red en el entorno de producción

Los servidores donde se encuentran instaladas las instancias del GDA eXcriba 2.0, están conectados a la Red de Área Local (*Local Area Network*, en inglés, abreviado LAN) de la universidad. El ancho de banda de la red en el centro de datos es de 1 Gb/s.

2.2.5 Configuraciones de *hardware* en el entorno de prueba

El entorno de prueba se ha configurado en el laboratorio 205 del Centro de Informatización Universitaria de la Facultad 1. Se ha instalado el sistema con las mismas características que en el entorno de producción.

Se utilizan dos servidores, uno que aloja la base de datos y otro que contiene el cliente web, el núcleo del sistema y el almacén de contenidos.

Tabla 6: Configuraciones de *hardware* en el entorno de prueba.

Recurso	Servidor del GDA eXcriba 2.0	Servidor de base de datos
Sistema Operativo	Debian 6.0	Debian 6.0
RAM	4.0 GB	2.0 GB
CPU	Intel (R) Core 2 DUO 2.2 GHz	Intel (R) Core 2 DUO 2.2 GHz
Arquitectura	64 bits	32 bits
Ancho de banda	10 Mb/s	10 Mb/s
Disco	Seagate 7 200 rpm 160 GB 300 GB/s	

2.2.6 Configuraciones de *software* en el entorno de prueba

Se ha instalado el GDA eXcriba 2.0 con características similares al entorno de producción. Se ha configurado el sistema con los parámetros de la Máquina Virtual de Java como se muestra:

```
JAVA_OPTS='-server -Xms2048m -Xmx2048m -XX:NewSize=256m -XX:NewRatio=2
      -XX:MaxPermSize=128m -Djava.awt.headless=true -XX:+UseConcMarkSweepGC
      -XX:+CMSIncrementalMode'
```

2.2.7 Configuraciones de red en el entorno de prueba

El ancho de banda configurado dentro del laboratorio es de 10/100 Mb/s.

2.2.8 Recursos para realizar las pruebas

Se utilizan dos máquinas para simular el entorno de producción en el laboratorio de prueba: la primera aloja el sistema y la segunda contiene la base de datos. Se usa la herramienta JMeter en su versión 2.9 para automatizar el proceso de pruebas.

2.3 Identificar los criterios de aceptación de rendimiento

Los objetivos de rendimiento constituyen el conjunto de criterios de aceptación que el equipo de desarrollo necesita satisfacer antes de ser liberado el producto, incluso esos criterios pueden ser reajustados bajo ciertas circunstancias (30). En la investigación se propone como objetivos de rendimiento para el GDA eXcriba 2.0: el tiempo de respuesta, la capacidad de procesamiento, el por ciento de uso del CPU y la utilización de la memoria RAM.

Para determinar los criterios de aceptación de rendimiento se tuvo en consideración la opinión del equipo de desarrollo acerca de cómo consideraban que debería funcionar el sistema basado en sus experiencias

investigando en temas relacionados con la gestión documental, por el tiempo usando y desarrollando el sistema y el rol que ejercen en el proyecto. Por otra parte, se tuvo en cuenta las experiencias acumuladas por clientes del GDA eXcriba 2.0 tales como:

- ENIA: Empresa Nacional de Investigaciones Aplicadas.
- DCH: Empresa de Diseño de Ciudad de La Habana.
- Oficina del Consejo de estado.
- MIC: Ministerio de Informática y las Comunicaciones.
- Contraloría General de la República.
- Aduana General de la República.
- Radio Reloj
- Archivo General de la Nación de Venezuela.

En la siguiente tabla se relacionan las principales características de los miembros del equipo:

Tabla 7: Caracterización del equipo de desarrollo.

Miembro	PF	GC	CD	R	AGD	AS
Misael Fonseca Mata	Ingeniero en Ciencias Informáticas	-	Instructor	Jefe de Dpto.	6	6
Dayelis Blanco Hernández	Ingeniera en Ciencias Informáticas	-	Instructor	Jefa de Proyecto	5	5
Marcel Sánchez Góngora	Ingeniero en Ciencias Informáticas	-	-	Responsable Soporte Tecnológico	5	5
Michel David Suárez	Ingeniero en Ciencias Informáticas	-	-	Desarrollador	6	6
Reinier Elejalde Chacón	Ingeniero en Ciencias Informáticas	-	-	Desarrollador	6	6
Pedro Rodríguez Samon	Ingeniero en Ciencias Informáticas	-	Instructor	Desarrollador	5	5
Lizandra Candelario Rodríguez						
Laritza Cabrera Barroso	Ingeniera en Ciencias Informáticas	Msc.	Asistente	Analista	4	2
Adrian Cid Almaguer	Ingeniero en Ciencias Informáticas	Msc.	Asistente	Desarrollador	7	3

PF: Profesión
GC: Grado Científico
CD: Categoría Docente
R: Rol
AGD: Años de experiencia investigando temas relacionados a la gestión documental
AS: Años de experiencia trabajando con el sistema

A partir de las razones anteriormente expuestas se determina:

- El promedio de tiempo de respuesta de las peticiones no debe exceder de 4s.
- El consumo promedio de CPU no debe exceder el 70%.

Se consideró que el criterio relacionado con la capacidad de procesamiento es posible determinarlo a partir de las pruebas al sistema, pues es una característica propia de aplicación y menos percibida por el usuario.

2.4 Planificar y diseñar las pruebas

No es posible definir métricas de rendimiento para cada posible uso del sistema; de ahí que es necesario enfocarse en los escenarios más significativos o representativos (31). Por tanto, para diseñar las pruebas se tuvo en cuenta los casos de prueba críticos a consideración del equipo de desarrollo. “Un **caso de prueba** especifica una forma de probar el sistema, incluyendo la entrada o resultado con la que se ha de probar y las condiciones bajo las que se ha de probar” (32). Los casos de prueba críticos identificados son Adicionar contenido y Eliminar contenido. Es importante que el caso de prueba describa una ruta simple para ejecutar una acción, pues añadir rutas condicionales, como el tratamiento de errores por acciones inválidas del usuario, puede provocar un aumento del tiempo que le tome al sistema en evaluar los posibles resultados.

Los casos de prueba se implementaron usando la herramienta JMeter; para ello se utilizaron *scripts* y escenarios de prueba. Un ***script* de prueba** es un programa creado por el especialista que realiza las pruebas de rendimiento para realizar todas las acciones que incluye un caso de prueba. Los **escenarios de prueba** describen la manera en que se deben ejecutar los *scripts* (33).

Otro aspecto fundamental a delimitar en esta etapa son los objetivos que se persiguen al realizar las pruebas de rendimiento sobre el sistema.

2.4.1 Objetivos de las pruebas de rendimiento sobre el GDA eXcriba 2.0

- Definir las métricas de rendimiento en el entorno de producción para cada uno de los casos de prueba críticos.
- Verificar que la aplicación web puede soportar la carga de trabajo definida por el equipo de desarrollo del sistema a probar.
- Identificar cuál es el número máximo de usuarios concurrentes que soportará el sistema sin que se degrade su rendimiento.
- Determinar el tiempo de respuesta de las páginas ante peticiones realizadas al sistema.

2.4.2 Diseño de los casos de prueba

A continuación se muestra el diseño de cada uno de los casos de prueba identificados para realizar las pruebas. Se tuvo en cuenta el diseño propuesto por el Ingeniero en Sistemas de Computadoras Stuart Moncrieff (33). Para la descripción de cada uno de los campos se puede consultar el **Anexo 3: Modelo para el diseño de casos de prueba de rendimiento.**

Tabla 8: Caso de prueba Adicionar contenido.

Caso de prueba:	Adicionar contenido
Descripción:	Este caso de prueba simula las acciones que puede realizar un usuario para adicionar un contenido al sistema.
Requisitos:	El usuario tiene que tener el rol de contribuidor, colaborador, coordinador o cooperador en el espacio donde desea adicionar el contenido. {nombre_usuario}: tiene que ser único para cada usuario en el sistema. {contraseña}: tiene que ser válida para el nombre de usuario especificado. Los campos {archivo} y {tipo_contenido} son obligatorios para adicionar el contenido.

No.	**Descripción**	**Resultado esperado**	**Nombre**	**Tiempo**
01	Abrir el sistema mediante el navegador web. Introducir los campos requeridos para la autenticación {nombre_ usuario} y {contraseña}.	Se muestra la página principal del sistema.	Autenticar_usuario	5
02	Seleccionar el espacio donde se adicionará el contenido.	Se muestra un formulario con los campos requeridos.	Buscar_espacio	20
03	Insertar los campos requeridos: {titulo}, {archivo}, {tipo_contenido}, {descripción}. Presionar el botón Adicionar.	Se muestra una notificación. Se muestra el último espacio accedido.	Adicionar_contenido	15

Tabla 9: Caso de prueba Eliminar contenido.

Caso de prueba:	Eliminar contenido
Descripción:	Este caso de prueba simula las acciones que puede realizar un usuario para eliminar contenidos en el sistema.
Requisitos:	El usuario tiene que tener el rol de coordinador o cooperador en el espacio donde desea eliminar el contenido o sobre el contenido. {nombre_usuario}: tiene que ser único para cada usuario en el sistema. {contraseña}: tiene que ser válida para el nombre de usuario especificado.

No.	**Descripción**	**Resultado esperado**	**Nombre**	**Tiempo**

01	Abrir el sistema mediante el navegador web. Introducir los campos requeridos para la autenticación {nombre_ usuario} y {contraseña}.	Se muestra la página principal del sistema.	Autenticar_usuario	5
02	Insertar en el campo de búsqueda el nombre del espacio donde se encuentra el contenido a eliminar.	Se muestra una página con los resultados coincidentes al parámetro de búsqueda.	Buscar_espacio	2
03	Seleccionar el espacio y en el contenido que se desea la opción eliminar.	Se muestra un mensaje de confirmación de la acción. Se muestra el último espacio accedido.	Eliminar_contenido	2
04	Seleccionar la opción para cerrar la sesión del usuario.	Se muestra la página para la autenticación en el sistema.	Cerrar_sesion	0

2.5 Configurar el entorno de prueba

Como parte de las acciones desarrolladas en esta etapa, se incluyó en JMeter un módulo de complementos con nuevas funcionalidades de gran utilidad para extraer la información relacionada con la utilización de los recursos del servidor donde se encuentra instalado el GDA eXcriba 2.0 durante el proceso de prueba. Además, se configuró y se instaló JMeter en 4 estaciones de trabajo para ejecutar las pruebas distribuidas y estresar el sistema.

2.6 Implementar el diseño de las pruebas

Se verificó que el entorno de prueba coincidiera con la configuración y se implementó el diseño de prueba en JMeter. Para ello, se creó un plan de pruebas, se le adicionó el elemento grupo de hilos que simula la carga en la herramienta, se grabaron los escenarios diseñados en el caso de prueba mediante el elemento Servidor *Proxy* HTTP y se añadieron los elementos que generaron los reportes durante el proceso como el informe agregado, el reporte resumen, el árbol de resultados y el recolector de métricas de los recursos de *hardware*.

Resultaron de gran utilidad otros elementos importantes que proporciona JMeter para la elaboración de los planes de prueba; tal es el caso de los contadores para iterar sobre los ficheros de prueba que se añadieron y se eliminaron en el sistema. También se utilizaron variables globales para definir una característica genérica de los archivos que se gestionan en las pruebas y además, elementos de gestión de *cache* y de *cookies* debido a que la aplicación los requiere. Para este último caso se hizo necesario el uso del elemento de configuración *"BeanShell PreProcessor"* debido a que el GDA eXcriba 2.0 genera *cookies* menos

convencionales que son requeridas para ejecutar determinadas funcionalidades y el elemento de gestión de *cookies* no las reconoce.

Otro aspecto a destacar fue la utilización del elemento de extracción de expresiones regulares, contribuyó por un lado a la extracción de las *cookies* que no se reconocían cuando el usuario se autenticaba en el sistema y además, se requirió durante la ejecución del caso de prueba Eliminar contenido. Debido a que en el sistema no es posible que varios usuarios eliminen un mismo documento, se hizo necesario crear una expresión regular que permite obtener todos los contenidos de un espacio que cumplen con una característica definida, para de ahí seleccionar el identificador de cada elemento a eliminar por un usuario.

A continuación se ilustra la implementación en la herramienta del diseño previsto para el caso de prueba Eliminar contenido:

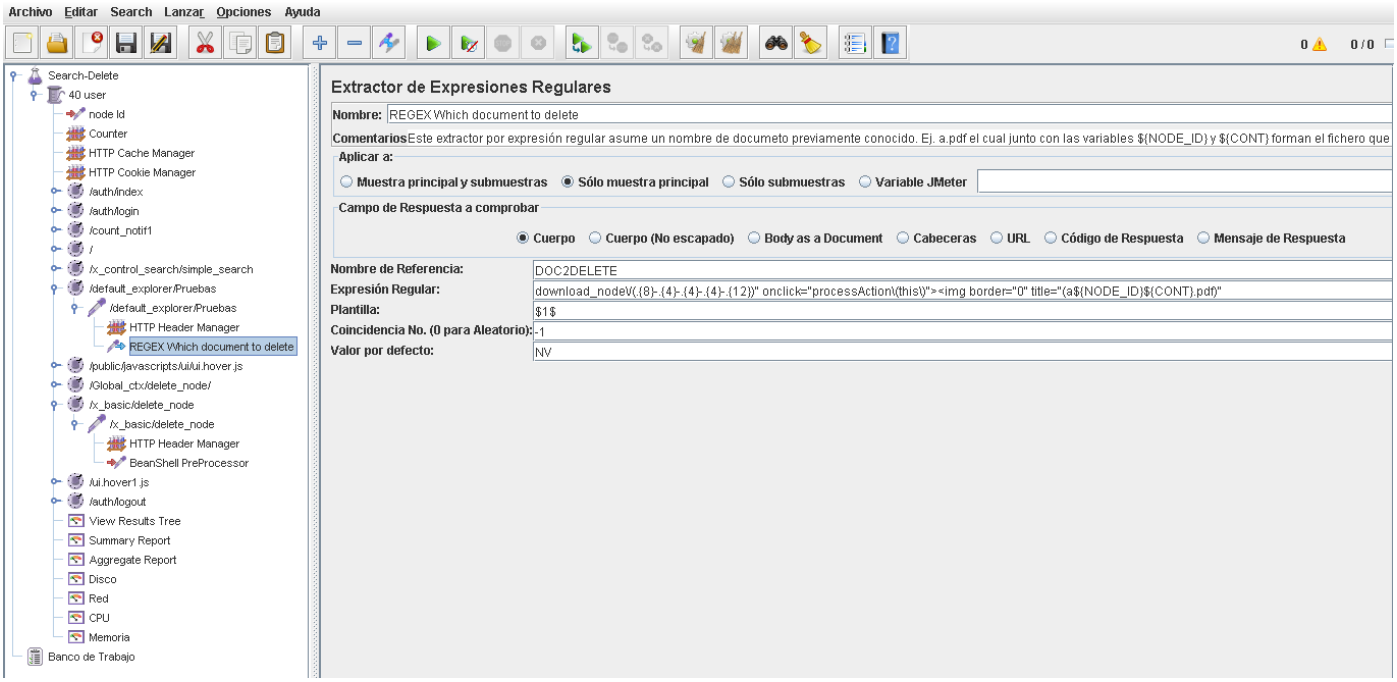

Figura 5: Implementación del caso de prueba Eliminar contenido.

2.7 Ejecución de las pruebas en el entorno de prueba

En este epígrafe se ilustran los principales resultados arrojados durante las pruebas para cada uno de los casos de prueba.

Para determinar el uso promedio del CPU se utiliza la siguiente ecuación:

$$\%\ Prom\ CPU = \frac{\sum_1^c \left(\frac{\sum_1^n V}{n}\right)}{c}$$

Donde:

- *V*: valor en por ciento de uso de un núcleo del CPU.
- *n*: cantidad de veces que fue descrito el uso de un núcleo del CPU.
- *c*: cantidad de núcleos del CPU.

El resultado de la ecuación se obtiene a partir de los parámetros de uso del CPU que registra JMeter para describir el comportamiento de este recurso durante el tiempo de prueba.

Caso de prueba Adicionar contenido

Para la ejecución del caso de prueba Adicionar contenido se simula la carga para un usuario y 40 usuarios respectivamente. En el primer caso, como se puede apreciar en la tabla siguiente, se obtiene un tiempo promedio de respuesta de las peticiones de 745 ms, durante el cual el sistema logra procesar como promedio 0.5 peticiones por segundo y 9.5 KB de información por segundo.

Tabla 10: Resultado para el caso de prueba Adicionar contenido en el entorno de prueba.

Nro. de usuarios	Tiempo de respuesta	Capacidad de procesamiento	Capacidad de procesamiento	%Err
1	745 ms	0.5/s	9.5 KB/s	0.0
40	11195 ms	11.4/s	2.2 KB/s	0.0

Mediante el monitoreo de los recursos de RAM y de CPU se aprecia que el comportamiento de la memoria se mantiene estable y de la capacidad total de 4 GB no excede los 2.15 GB de uso; mientras que el por ciento promedio de utilización del CPU es aproximadamente el 25 % durante la ejecución de las pruebas. Para una mejor comprensión, ver el **Anexo 4: Utilización de los recursos en el entorno de prueba.**

Cuando se aumenta la carga en el sistema para 40 usuarios, se obtiene un tiempo promedio de respuesta de las peticiones de 11195 ms, durante el cual la aplicación logra procesar como promedio 11.4 peticiones por segundo y 2.2 KB de información por segundo; como se muestra en la tabla superior.

Mediante el monitoreo de los recursos de RAM y de CPU se aprecia que el comportamiento de la memoria asciende de manera irregular sin sobrepasar los 3.78 GB de uso; mientras que el por ciento promedio de utilización del CPU es aproximadamente el 96.8 % durante la ejecución de las pruebas. Para una mejor comprensión, ver el **Anexo 4: Utilización de los recursos en el entorno de prueba.**

Teniendo en consideración lo explicado, se resume que en este caso de prueba el valor promedio de tiempo de respuesta y el por ciento promedio de uso del CPU para un usuario satisfacen los criterios de aceptación de rendimiento definidos con anterioridad; no así para 40 usuarios donde ambas métricas sobrepasan los criterios. Ver el epígrafe **2.3 Identificar los criterios de aceptación de rendimiento.**

Caso de prueba Eliminar contenido

Para la ejecución del caso de prueba Eliminar contenido se simula la carga para un usuario y 40 usuarios respectivamente. En el primer caso, como se puede apreciar en la tabla siguiente, se obtiene un tiempo promedio de respuesta de las peticiones de 983 ms, durante el cual el sistema logra procesar como promedio 1.0 peticiones por segundo y 23.6 KB de información por segundo.

Tabla 11: Resultado para el caso de prueba Eliminar contenido en el entorno de prueba.

Nro. de usuarios	Tiempo de respuesta	Capacidad de procesamiento	Capacidad de procesamiento	%Err
1	983 ms	1.0/s	23.6 KB/s	0.0
40	20900 ms	0.8/s	1.8 KB/s	0.0

Mediante el monitoreo de los recursos de RAM y de CPU se aprecia que el comportamiento de la memoria se mantiene estable y de la capacidad total de 4 GB no excede los 1.56 GB de uso; mientras que el por ciento promedio de utilización del CPU es aproximadamente el 64 % durante la ejecución de las pruebas. Para una mejor comprensión, ver las figuras 11 y 12 en el **Anexo 4: Utilización de los recursos en el entorno de prueba.**

Cuando se aumenta la carga en el sistema para 40 usuarios, se obtiene un tiempo promedio de respuesta de las peticiones de 20900 ms, durante el cual la aplicación logra procesar como promedio 0.8 peticiones por segundo y 1.8 KB de información por segundo; como se muestra en la tabla superior.

Mediante el monitoreo de los recursos de RAM y de CPU se aprecia que el comportamiento de la memoria asciende sin sobrepasar los 2.49 GB de uso; mientras que el por ciento promedio de utilización del CPU es aproximadamente el 97.3 % durante la ejecución de las pruebas. Para una mejor comprensión, ver las figuras 13 y 14 en el **Anexo 4: Utilización de los recursos en el entorno de prueba.**

Teniendo en consideración lo explicado, se resume que en este caso de prueba el valor promedio de tiempo de respuesta y el por ciento promedio de uso del CPU para un usuario satisfacen los criterios de aceptación de rendimiento definidos con anterioridad; no así para 40 usuarios donde ambas métricas sobrepasan los criterios. Ve el epígrafe **2.3 Identificar los criterios de aceptación de rendimiento.**

Conclusiones parciales

- La caracterización del entorno de producción, incluyendo la configuraciones de *software*, *hardware* y de red, permitieron identificar los principales elementos que describen el entorno físico de prueba, así como las herramientas y los recursos que utilizó el equipo de prueba durante el proceso.
- La consulta realizada al equipo de desarrollo sobre el rendimiento del GDA eXcriba 2.0 y las experiencias acumuladas por el despliegue del sistema en diferentes entidades contribuyó a identificar los criterios de aceptación de rendimiento que debe cumplir el GDA eXcriba 2.0 durante el proceso de prueba.
- El diseño de los casos de prueba, las configuraciones realizadas al entorno de prueba y la implementación del diseño previsto permitió dejar el entorno de prueba listo para ejecutar las pruebas de rendimiento, lo que facilitó la comprensión del proceso y la disminución del tiempo de ejecución.
- Mediante las pruebas de rendimiento realizadas al GDA eXcriba 2.0 en el entorno de prueba se obtuvieron las métricas de rendimiento: tiempo de respuesta, capacidad de procesamiento y utilización de la RAM y el CPU, contribuyendo a la caracterización del desempeño del sistema en ese contexto.

Capítulo III: Métricas de rendimiento

En el presente capítulo se continúa con la ejecución de las pruebas en el entorno de producción. Se determinan las métricas iniciales del GDA eXcriba 2.0 en la Red de Centros de la Universidad de las Ciencias Informáticas. Además se propone los parámetros de ajuste de la Máquina Virtual de Java y mediante las pruebas de rendimiento se obtienen las métricas que actualmente exhibe el sistema.

3.1 Ejecución de las pruebas en el entorno de producción

Caso de prueba Adicionar contenido en el Centro CISED

Para la ejecución del caso de prueba se simula la carga para 1, 10, 20 y 40 usuarios respectivamente. Como se puede apreciar en la tabla siguiente el tiempo promedio de respuesta de las peticiones va aumentando aproximadamente el doble de su valor a medida que se incrementa la carga; no así la capacidad de procesamiento, en la cual se puede observar que aumenta bruscamente de 1 a 20 usuarios, pero disminuye para 40 usuarios concurrentes.

Tabla 12: Resultado para el caso de prueba Adicionar contenido en el Centro CISED.

Nro. de usuarios	Tiempo de respuesta	Capacidad de procesamiento (tps)	Capacidad de procesamiento (KB/s)	%Err
1	806 ms	0.5/s	3.2 KB/s	0.0
10	2213 ms	2.8/s	30.8 KB/s	0.0
20	4112 ms	3.6/s	35.0 KB/s	0.0
40	9894 ms	3.2/s	33.0 KB/s	0.0

Mediante el monitoreo de los recursos de RAM y de CPU se aprecia que la utilización de la memoria aumenta con el incremento de la carga, aunque su comportamiento no excede los 1.97 GB, siendo su capacidad total de 2 GB. Por otra parte, el por ciento promedio de utilización del CPU se incrementa a medida que aumenta la carga, obteniéndose los resultados 17%, 69%, 83% y 92% respectivamente. Para ilustrar lo explicado, ver el **Anexo 6: Utilización de los recursos en el Centro CISED.**

Teniendo en cuenta el epígrafe **2.3 Identificar los criterios de aceptación de rendimiento** se observa que el valor promedio de tiempo de respuesta y el por ciento promedio de uso del CPU, a partir de los 20 usuarios concurrentes, sobrepasan los criterios definidos.

Caso de prueba Eliminar contenido en el Centro CISED

Para la ejecución del caso de prueba se simula la carga para 1, 10, 20 y 40 usuarios respectivamente. Como se puede apreciar en la tabla siguiente el tiempo promedio de respuesta de las peticiones va aumentando su valor a medida que se incrementa la carga; no así la capacidad de procesamiento, en la cual se puede observar que aumenta bruscamente de 1 a 10 usuarios, disminuye a partir de 20 usuarios concurrentes.

Tabla 13: Resultado para el caso de prueba Elimina contenido en el Centro CISED.

Nro. de usuarios	Tiempo de respuesta	Capacidad de procesamiento (tps)	Capacidad de procesamiento (KB/s)	%Err
1	1078 ms	0.9/s	9.3 KB/s	0.0
10	2969 ms	3.1/s	38.7 KB/s	0.0
20	6476 ms	3.0/s	38.0 KB/s	0.0
40	13981 ms	2.8/s	30.0 KB/s	0.2

Mediante el monitoreo de los recursos de RAM y de CPU se aprecia que la utilización de la memoria aumenta con el incremento de la carga, aunque su comportamiento no excede los 1.96 GB, siendo su capacidad total de 2 GB. Por otra parte, el por ciento promedio de utilización del CPU se incrementa a medida que aumenta la carga, obteniéndose los resultados 21%, 83%, 86% y 88% respectivamente. Para ilustrar lo explicado, ver el **Anexo 6: Utilización de los recursos en el Centro CISED.**

Teniendo en cuenta el epígrafe **2.3 Identificar los criterios de aceptación de rendimiento** se observa que el valor promedio de tiempo de respuesta a partir de los 20 usuarios concurrentes sobrepasa el criterio definido; mientras que el por ciento promedio de uso del CPU, lo sobrepasa a partir de los 10 usuarios concurrentes.

Caso de prueba Adicionar contenido en el Centro CDAE

Para la ejecución del caso de prueba se simula la carga para 1, 10, 20 y 40 usuarios respectivamente. Como se puede apreciar en la tabla siguiente el tiempo promedio de respuesta de las peticiones y la capacidad de procesamiento van aumentando su valor a medida que se incrementa la carga.

Tabla 14: Resultado para el caso de prueba Adicionar contenido en el Centro CDAE.

Nro. de usuarios	Tiempo de respuesta	Capacidad de procesamiento (tps)	Capacidad de procesamiento (KB/s)	%Err
1	830 ms	0.7/s	2.8 KB/s	0.0
10	1855 ms	3.1/s	24.8 KB/s	0.0
20	4493 ms	3.2/s	25.0 KB/s	0.0
40	9748 ms	3.3/s	26.2 KB/s	0.0

Mediante el monitoreo de los recursos de RAM y de CPU se aprecia que la utilización de la memoria aumenta con el incremento de la carga, aunque su comportamiento no excede los 1.97 GB, siendo su capacidad total de 2 GB. Por otra parte, el por ciento promedio de utilización del CPU se incrementa a medida que aumenta la carga, obteniéndose los resultados 17.6%, 65%, 92.1% y 93.5% respectivamente. Para ilustrar lo explicado, ver el **Anexo 7: Utilización de los recursos en el Centro CDAE.**

Teniendo en cuenta el epígrafe **2.3 Identificar los criterios de aceptación de rendimiento** se observa que el valor promedio de tiempo de respuesta y el por ciento promedio de uso del CPU a partir de los 20 usuarios concurrentes sobrepasan los criterios definidos.

Caso de prueba Eliminar contenido en el Centro CDAE

Para la ejecución del caso de prueba se simula la carga para 1, 10, 20 y 40 usuarios respectivamente. Como se puede apreciar en la tabla siguiente el tiempo promedio de respuesta de las peticiones va aumentando su valor a medida que se incrementa la carga; no así la capacidad de procesamiento, en la cual se puede observar que aumenta bruscamente de 1 a 20 usuarios y disminuye para 40 concurrentes.

Tabla 15: Resultado para el caso de prueba Eliminar contenido en el Centro CDAE.

Nro. de usuarios	Tiempo de respuesta	Capacidad de procesamiento (tps)	Capacidad de procesamiento (KB/s)	%Err
1	1286 ms	0.7/s	7.4 KB/s	0.0
10	3997ms	2.2/s	32.3 KB/s	1.9
20	6820 ms	2.6/s	35.0 KB/s	2.6
40	18508 ms	2.0/s	30.0 KB/s	1.0

Mediante el monitoreo de los recursos de RAM y de CPU se aprecia que la utilización de la memoria aumenta con el incremento de la carga, aunque su comportamiento no excede los 1.90 GB, siendo su capacidad total de 2 GB. Por otra parte, el por ciento promedio de utilización del CPU se incrementa a medida que aumenta la carga, siendo los resultados 19%, 92.6%, 92.7% y 94% respectivamente. Para ilustrar lo explicado, ver el **Anexo 7: Utilización de los recursos en el Centro CDAE.**

Teniendo en cuenta el epígrafe **2.3 Identificar los criterios de aceptación de rendimiento** se observa que el valor promedio de tiempo de respuesta a partir de los 20 usuarios concurrentes sobrepasa el criterio definido; mientras que el por ciento promedio de uso del CPU, lo sobrepasa a partir de los 10 usuarios concurrentes.

Caso de prueba Adicionar contenido en el Centro CEGEL

Para la ejecución del caso de prueba se simula la carga para 1, 10, 20 y 40 usuarios respectivamente. Como se puede apreciar en la tabla siguiente el tiempo promedio de respuesta de las peticiones y la capacidad de procesamiento van aumentando su valor a medida que se incrementa la carga.

Tabla 16: Resultado para el caso de prueba Adicionar contenido en el Centro CEGEL.

Nro. de usuarios	Tiempo de respuesta	Capacidad de procesamiento (tps)	Capacidad de procesamiento (KB/s)	%Err
1	907 ms	0.5/s	9.9 KB/s	0.0
10	4403 ms	1.0/s	39.7 KB/s	0.0
20	14855 ms	1.2/s	41.0 KB/s	0.0
40	23024 ms	2.1/s	50.0 KB/s	0.0

Mediante el monitoreo de los recursos de RAM y de CPU se aprecia que la utilización de la memoria aumenta con el incremento de la carga, aunque su comportamiento no excede los 3.3 GB, siendo su capacidad total de 8 GB. Por otra parte, el por ciento promedio de utilización del CPU se incrementa a medida que aumenta la carga, siendo los resultados 17%, 68.8%, 95.3% y 96% respectivamente. Para ilustrar lo explicado, ver el **Anexo 8: Utilización de los recursos en el Centro** CEGEL**.**

Teniendo en cuenta el epígrafe **2.3 Identificar los criterios de aceptación de rendimiento** se observa que el valor promedio de tiempo de respuesta y el por ciento promedio de uso del CPU a partir de los 20 usuarios concurrentes sobrepasan los criterios definidos.

Caso de prueba Eliminar contenido en el Centro CEGEL

Para la ejecución del caso de prueba se simula la carga para 1, 10, 20 y 40 usuarios respectivamente. Como se puede apreciar en la tabla siguiente el tiempo promedio de respuesta de las peticiones va aumentando su valor a medida que se incrementa la carga; no así la capacidad de procesamiento, en la cual se puede observar que aumenta de 1 a 10 usuarios, disminuye a partir de 20 concurrentes. Además es apreciable que este caso de prueba para 20 y 40 usuarios se ha ejecutado con error debido a, según las trazas de la herramienta, a problemas de conexión. Durante la prueba se comprobó que las peticiones que afectaban el éxito del proceso fueron específicamente algunas acciones para eliminar un contenido; aclarar que todas

no resultaron fallidas, algunas llegaron a ejecutarse.

Tabla 17: Resultado para el caso de prueba Eliminar contenido en el Centro CEGEL.

Nro. de usuarios	Tiempo de respuesta	Capacidad de procesamiento (tps)	Capacidad de procesamiento (KB/s)	%Err
1	1718 ms	0.6/s	10.9 KB/s	0.0
10	6415 ms	0.8/s	53.6 KB/s	0.0
20	40718ms	0.6/s	48.2 KB/s	0.2
40	54947 ms	0.7/s	50.0 KB/s	0.3

Mediante el monitoreo de los recursos de RAM y de CPU se aprecia que la utilización de la memoria aumenta con el incremento de la carga, aunque su comportamiento no excede los 3.5 GB, siendo su capacidad total de 8 GB. Por otra parte, el por ciento promedio de utilización del CPU se incrementa a medida que aumenta la carga, siendo los resultados 19%, 74.5%, 85.3% y 92% respectivamente. Para ilustrar lo explicado, ver el **Anexo 8: Utilización de los recursos en el Centro** CEGEL**.**

Teniendo en cuenta el epígrafe **2.3 Identificar los criterios de aceptación de rendimiento** se observa que el valor promedio de tiempo de respuesta y el por ciento promedio de uso del CPU a partir de los 10 usuarios concurrentes sobrepasan los criterios definidos.

3.2 Ajuste de los parámetros de la Máquina Virtual de Java

Después de realizadas las pruebas de rendimiento al GDA eXcriba 2.0 en el entorno de producción se decide ajustar los parámetros de la Máquina Virtual de Java sobre la cual funciona el sistema con el propósito de mejorar su desempeño. A continuación se describen aspectos importantes que son necesarios conocer sobre la JVM y se proponen las configuraciones.

Cuando se inicia un programa en Java, la Máquina Virtual obtiene parte de la memoria del Sistema Operativo, la que usa para realizar todas sus funciones; una porción de esa memoria es conocida como memoria *Heap* de Java. Los objetos se crean en la memoria *Heap* y el Recolector de Basura (en inglés *Garbage Collector*) es el encargado de eliminar los objetos en desuso y devolverle el espacio en memoria al *Heap* en Java.

El *Heap* del Recolector de Basura se divide en tres regiones fundamentales: la Nueva Generación (en inglés *New Generation*), la Antigua Generación (en inglés *Old Generation*) y la Generación Permanente (en inglés *Perm Generation*). La primera región almacena los objetos creados por la aplicación durante su ejecución y para mantenerlos con vida el Recolector de Basura los mueve a la Antigua Generación. La región Permanente es donde la Máquina Virtual de Java almacena los metadatos asociados a las clases y métodos y los detalles a nivel de clases (34).

Otro recurso importante lo constituye la pila (en inglés *stack*), que pudiera confundirse con el término *Heap*, mas tienen significados y son utilizados con propósitos diferentes. La pila, a diferencia del *Heap*, es utilizada para almacenar variables locales y llamadas de funciones; debe ser configurada con un tamaño mucho menor que el *Heap* ya que cada hilo en Java tiene su propia pila y un tamaño excesivo pudiera provocar que se genere la excepción `java.lang.StackOverFlowError` (35).

Para cada Centro se propone:

```
JAVA_OPTS="-XX:MaxPermSize=256m -Xms1024m -Xmx1024m -server
   -XX:NewSize=256m -XX:NewRatio=2 -XX:+UseConcMarkSweepGC
   -XX:CMSInitiatingOccupanceyFraction -XX:+CMSIncrementalMode"
```

Los detalles de cada uno de los parámetros configurados se pueden consultar en el **Anexo 2: Descripción de las opciones de la Máquina Virtual de Java.**

3.3 Ejecución de las pruebas en el entorno de producción después del ajuste

Caso de prueba Adicionar contenido en el Centro CISED

Para la ejecución del caso de prueba se simula la carga para 1, 10, 20 y 40 usuarios respectivamente. Como se puede apreciar en la tabla siguiente el tiempo promedio de respuesta de las peticiones y la capacidad de procesamiento van aumentando su valor a medida que se incrementa la carga.

Tabla 18: Resultado para el caso de prueba Adicionar contenido después del ajuste Centro CISED.

Nro. de usuarios	Tiempo de respuesta	Capacidad de procesamiento (tps)	Capacidad de procesamiento (KB/s)	%Err
1	706 ms	0.5/s	6.2 KB/s	0.0
10	1687 ms	3.2/s	42.3 KB/s	0.0
20	2250 ms	5.4/s	60.0 KB/s	0.0
40	4811 ms	5.9/s	56.3 KB/s	0.0

Mediante el monitoreo de los recursos de RAM y de CPU se aprecia que la utilización de la memoria aumenta con el incremento de la carga, aunque su comportamiento no excede los 1.96 GB, siendo su capacidad total de 2 GB. Por otra parte, el por ciento promedio de utilización del CPU se incrementa a medida que aumenta la carga, obteniéndose los resultados 16.2%, 61.9%, 65.3% y 84.5% respectivamente. Para ilustrar lo explicado, ver el **Anexo 9: Utilización de los recursos en el Centro CISED después del ajuste.**

Teniendo en cuenta el epígrafe **2.3 Identificar los criterios de aceptación de rendimiento** se observa que

el valor promedio de tiempo de respuesta y el por ciento promedio de uso del CPU a partir de los 40 usuarios concurrentes, sobrepasan los criterios definidos.

Caso de prueba Eliminar contenido en el Centro CISED

Para la ejecución del caso de prueba se simula la carga para 1, 10, 20 y 40 usuarios respectivamente. Como se puede apreciar en la tabla siguiente el tiempo promedio de respuesta de las peticiones va aumentando su valor a medida que se incrementa la carga; no así la capacidad de procesamiento, en la cual se puede observar que aumenta hasta 20 usuarios y disminuye a partir de 40 usuarios.

Tabla 19: Resultado para el caso de prueba Eliminar contenido después del ajuste Centro CISED.

Nro. de usuarios	Tiempo de respuesta	Capacidad de procesamiento (tps)	Capacidad de procesamiento (KB/s)	%Err
1	901 ms	1.1/s	17.0 KB/s	0.0
10	2411 ms	3.9/s	60.0 KB/s	0.0
20	3411 ms	5.2/s	70.8 KB/s	0.0
40	41768 ms	4.0/s	67.0 KB/s	0.0

Mediante el monitoreo de los recursos de RAM y de CPU se aprecia que la utilización de la memoria aumenta con el incremento de la carga, aunque su comportamiento no excede los 1.96 GB, siendo su capacidad total de 2 GB. Por otra parte, el por ciento promedio de utilización del CPU se incrementa a medida que aumenta la carga, obteniéndose los resultados 28.5%, 77.9%, 84.4% y 85.3% respectivamente. Para ilustrar lo explicado, ver el **Anexo 9: Utilización de los recursos en el Centro CISED después del ajuste.**

Teniendo en cuenta el epígrafe **2.3 Identificar los criterios de aceptación de rendimiento** se observa que el valor promedio de tiempo de respuesta a partir de los 40 usuarios y el por ciento promedio de uso del CPU a partir de 10 usuarios concurrentes, sobrepasan los criterios definidos.

Caso de prueba Adicionar contenido en el Centro CDAE

Para la ejecución del caso de prueba se simula la carga para 1, 10, 20 y 40 usuarios respectivamente. Como se puede apreciar en la tabla siguiente el tiempo promedio de respuesta de las peticiones y la capacidad de procesamiento van aumentando su valor a medida que se incrementa la carga.

Tabla 20: Resultado para el caso de prueba Adicionar contenido después del ajuste Centro CDAE.

Nro. de usuarios	Tiempo de respuesta	Capacidad de procesamiento (tps)	Capacidad de procesamiento (KB/s)	%Err
1	912 ms	0.5/s	6.6 KB/s	0.0
10	1805 ms	3.3/s	24.7 KB/s	0.0
20	3007 ms	4.4/s	30.0 KB/s	0.0
40	6393 ms	4.8/s	33.5 KB/s	0.0

Mediante el monitoreo de los recursos de RAM y de CPU se aprecia que la utilización de la memoria aumenta con el incremento de la carga, aunque su comportamiento no excede los 1.95 GB, siendo su capacidad total de 2 GB. Por otra parte, el por ciento promedio de utilización del CPU se incrementa a medida que aumenta la carga, obteniéndose los resultados 16.3%, 53.8%, 81.4% y 91.7% respectivamente. Para ilustrar lo explicado, ver el **Anexo 10: Utilización de los recursos en el Centro CDAE después del ajuste.**

Teniendo en cuenta el epígrafe **2.3 Identificar los criterios de aceptación de rendimiento** se observa que el valor promedio de tiempo de respuesta a partir de los 40 usuarios concurrentes, sobrepasa el criterio definido; mientras que el por ciento promedio de uso del CPU, a partir de los 20 usuarios concurrentes, excede el criterio.

Caso de prueba Eliminar contenido en el Centro CDAE

Para la ejecución del caso de prueba se simula la carga para 1, 10, 20 y 40 usuarios respectivamente. Como se puede apreciar en la tabla siguiente el tiempo promedio de respuesta de las peticiones y la capacidad de procesamiento van aumentando su valor a medida que se incrementa la carga. Este comportamiento se debe especialmente por los errores que se aprecian en la ejecución de las pruebas para 20 y 40 usuarios concurrentes, los cuales se produjeron debido a problemas de conexión, según los reportes que generó JMeter. Además se comprobó que las peticiones que provocaron las fallas en el caso de prueba eran las acciones específicas de eliminar un contenido, aunque no todas las peticiones resultaron fallidas.

Tabla 21: Resultado para el caso de prueba Eliminar contenido después del ajuste Centro CDAE.

Nro. de usuarios	Tiempo de respuesta	Capacidad de procesamiento (tps)	Capacidad de procesamiento (KB/s)	%Err
1	832 ms	1.1/s	17.69 KB/s	0.0
10	3055 ms	3.0/s	55.5 KB/s	0.0
20	5236 ms	3.6/s	57.0 KB/s	0.9
40	9468 ms	4.2/s	60.2 KB/s	2.9

Mediante el monitoreo de los recursos de RAM y de CPU se aprecia que la utilización de la memoria

aumenta con el incremento de la carga, aunque su comportamiento no excede los 1.94 GB, siendo su capacidad total de 2 GB. Por otra parte, el por ciento promedio de utilización del CPU se incrementa a medida que aumenta la carga, obteniéndose los resultados 25%, 86.4%, 95.5% y 94% respectivamente. Para ilustrar lo explicado, ver el **Anexo 10: Utilización de los recursos en el Centro CDAE después del ajuste.**

Teniendo en cuenta el epígrafe **2.3 Identificar los criterios de aceptación de rendimiento** se observa que el valor promedio de tiempo de respuesta a partir de los 20 usuarios y el por ciento promedio de uso del CPU a partir de los 10 usuarios concurrentes, sobrepasan los criterios definidos.

Caso de prueba Adicionar contenido en el Centro CEGEL

Para la ejecución del caso de prueba se simula la carga para 1, 10, 20 y 40 usuarios respectivamente. Como se puede apreciar en la tabla siguiente el tiempo promedio de respuesta de las peticiones va aumentando su valor a medida que se incrementa la carga; no así la capacidad de procesamiento, en la cual se puede observar que disminuye para 40 usuarios concurrentes.

Tabla 22: Resultado para el caso de prueba Adicionar contenido después del ajuste Centro CEGEL.

Nro. de usuarios	Tiempo de respuesta	Capacidad de procesamiento (tps)	Capacidad de procesamiento (KB/s)	%Err
1	774 ms	0.5/s	10.6 KB/s	0.0
10	3280 ms	1.4 /s	29.3 KB/s	0.0
20	8111 ms	2.5/s	30.0 KB/s	0.0
40	20217 ms	3.0/s	41.2 KB/s	0.0

Mediante el monitoreo de los recursos de RAM y de CPU se aprecia que la utilización de la memoria aumenta con el incremento de la carga, aunque su comportamiento no excede los 3 GB, siendo su capacidad total de 8 GB. Por otra parte, el por ciento promedio de utilización del CPU se incrementa a medida que aumenta la carga, obteniéndose los resultados 16%, 65%, 90% y 92% respectivamente. Para ilustrar lo explicado, ver el **Anexo 11: Utilización de los recursos en el Centro CEGEL después del ajuste.**

Teniendo en cuenta el epígrafe **2.3 Identificar los criterios de aceptación de rendimiento** se observa que el valor promedio de tiempo de respuesta y el por ciento promedio de uso del CPU a partir de los 20 usuarios concurrentes, sobrepasan los criterios definidos.

Caso de prueba Eliminar contenido en el Centro CEGEL

Para la ejecución del caso de prueba se simula la carga para 1, 10, 20 y 40 usuarios respectivamente. Como

se puede apreciar en la tabla siguiente el tiempo promedio de respuesta de las peticiones va aumentando su valor a medida que se incrementa la carga; no así la capacidad de procesamiento, en la cual se puede observar que disminuye su valor a partir de los 20 usuarios concurrentes. Este comportamiento se debe especialmente por los errores que se aprecian en la ejecución de las pruebas para 20 y 40 usuarios concurrentes, los cuales se produjeron debido a problemas de conexión, según los reportes que generó JMeter. Además se comprobó que las peticiones que provocaron las fallas en el caso de prueba eran las acciones específicas de eliminar un contenido, aunque no todas las peticiones resultaron fallidas.

Tabla 23: Resultado para el caso de prueba Eliminar contenido después del ajuste Centro CEGEL.

Nro. de usuarios	Tiempo de respuesta	Capacidad de procesamiento (tps)	Capacidad de procesamiento (KB/s)	%Err
1	1697 ms	0.7/s	16.5 KB/s	0.0
10	5236 ms	1.5/s	50.3 KB/s	0.0
20	28038 ms	1.1/s	48.8 KB/s	0.5
40	43561 ms	1.0/s	40.0 KB/s	0.7

Mediante el monitoreo de los recursos de RAM y de CPU se aprecia que la utilización de la memoria aumenta con el incremento de la carga, aunque su comportamiento no excede los 3.2 GB, siendo su capacidad total de 8 GB. Por otra parte, el por ciento promedio de utilización del CPU se incrementa a medida que aumenta la carga, obteniéndose los resultados 18%, 69%, 84.7% y 91.5% respectivamente. Para ilustrar lo explicado, ver el **Anexo 11: Utilización de los recursos en el Centro CEGEL después del ajuste.**

Teniendo en cuenta el epígrafe **2.3 Identificar los criterios de aceptación de rendimiento** se observa que el valor promedio de tiempo de respuesta a partir de los 10 usuarios y el por ciento promedio de uso del CPU, a partir de los 20 usuarios concurrentes, sobrepasan los criterios definidos.

3.3 Análisis comparativo de los resultados

En el presente epígrafe se realiza un análisis comparativo de los principales resultados obtenidos durante la investigación. En las gráficas se evidencia cómo se comportaron cada una de las métricas de rendimiento durante la ejecución de cada caso de prueba, antes y después de ser ajustados los parámetros de la Máquina Virtual de Java.

3.3.1 Caso de prueba Adicionar contenido:

Las figuras 6, 7 y 8 muestran el comportamiento del tiempo de respuesta para 1, 10, 20 y 40 usuarios concurrentes en cada Centro. Se observa que luego de ser ajustados los parámetros de la JVM disminuyeron los tiempos promedio de respuesta en cada Centro. En CISED y CDAE los tiempos de

respuesta cumplen con el criterio de aceptación de rendimiento hasta 20 usuarios concurrentes, no así en CEGEL que pese a la mejora, aún los resultados están muy por encima de lo esperado.

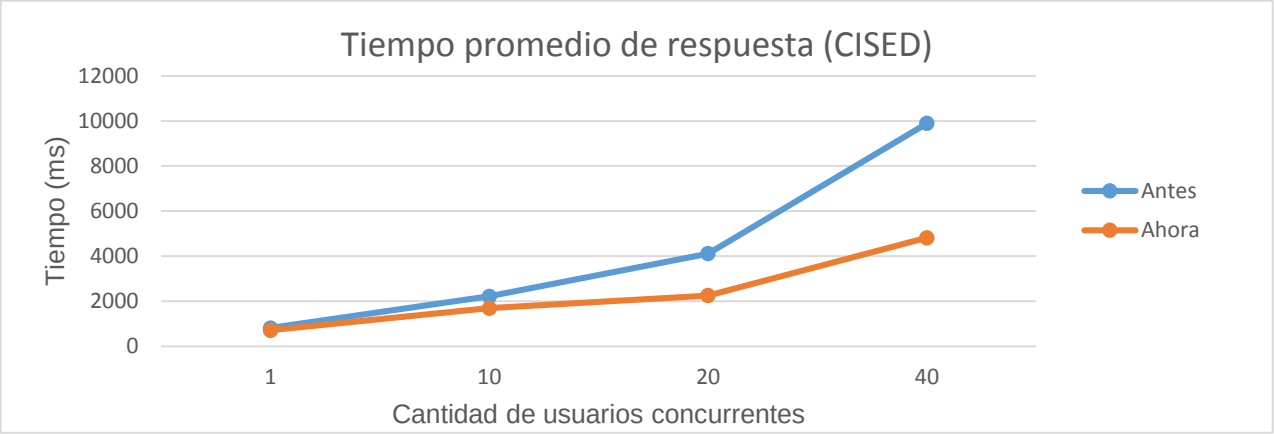

Figura 7: Tiempo promedio de respuesta en el Centro CISED.

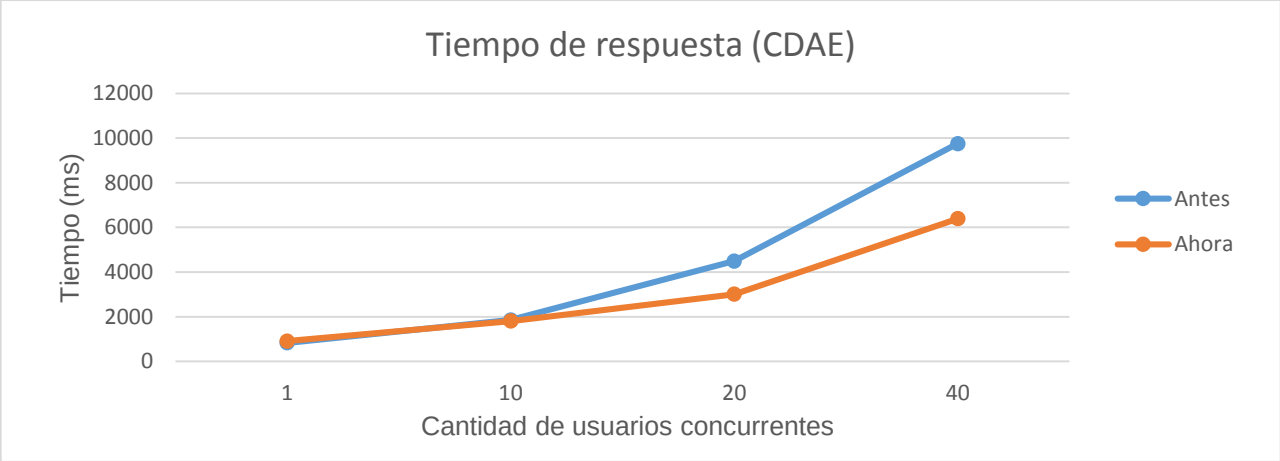

Figura 6: Tiempo promedio de respuesta en el Centro CDAE.

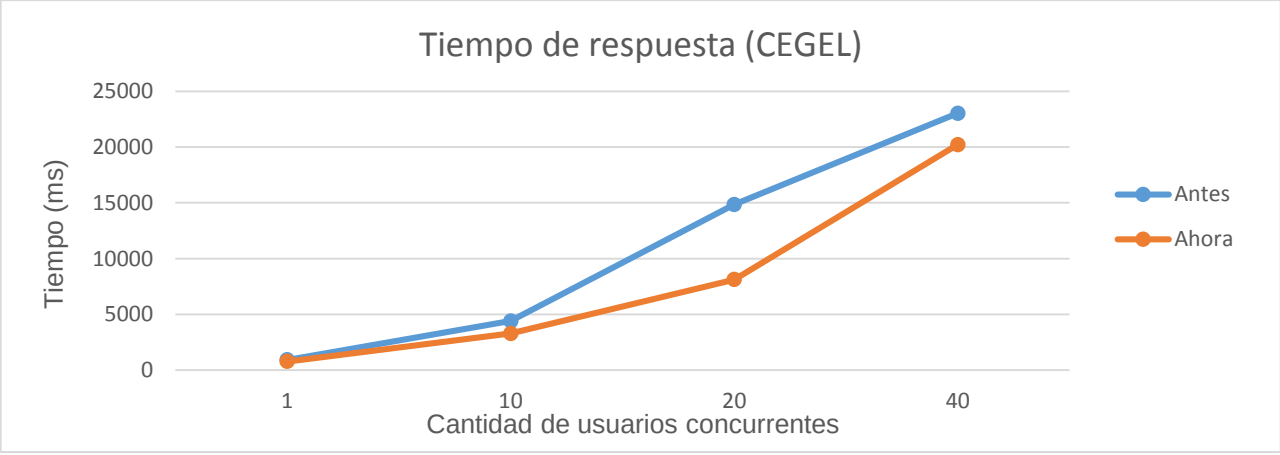

Figura 8: Tiempo promedio de respuesta en el Centro CEGEL.

Las figuras 9, 10, 11 muestran el comportamiento de la capacidad de procesamiento del sistema (en tps) para 1, 10, 20 y 40 usuarios concurrentes en cada Centro. Se observa que luego de ser ajustados los parámetros de la JVM aumenta dicha capacidad, aunque no se considera que estos valores sean los óptimos del sistema, pues son relativamente bajos.

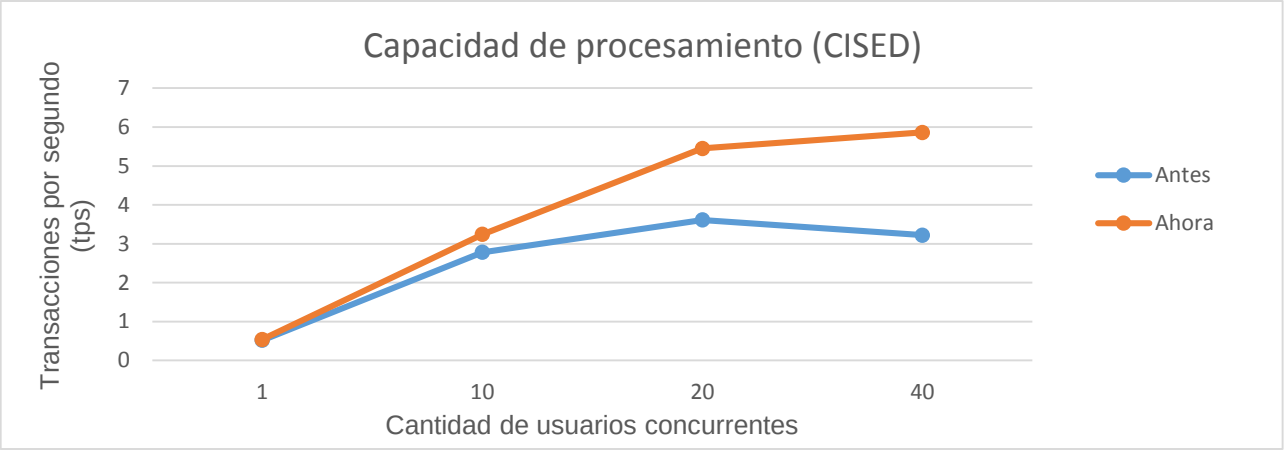

Figura 9: Capacidad de procesamiento promedio en el Centro CISED.

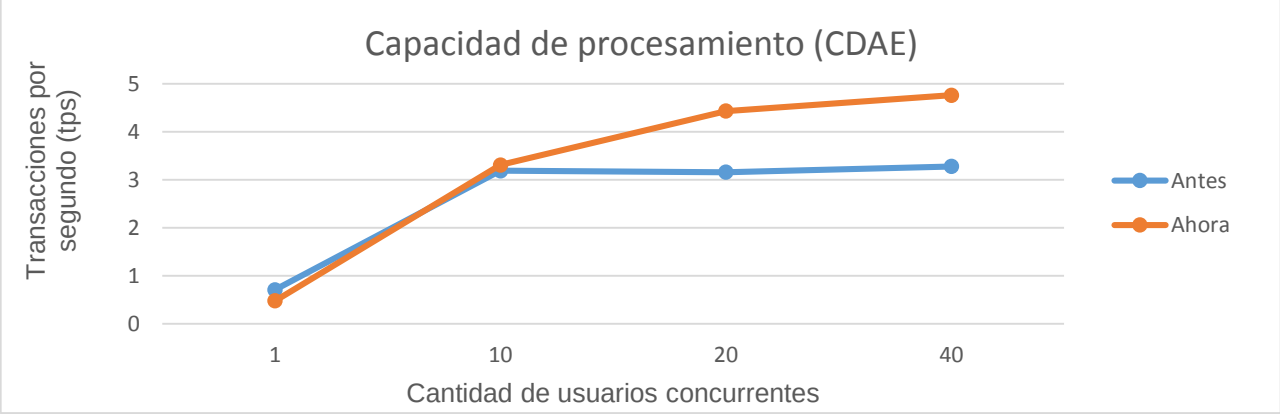

Figura 10: Capacidad de procesamiento promedio en el Centro CDAE.

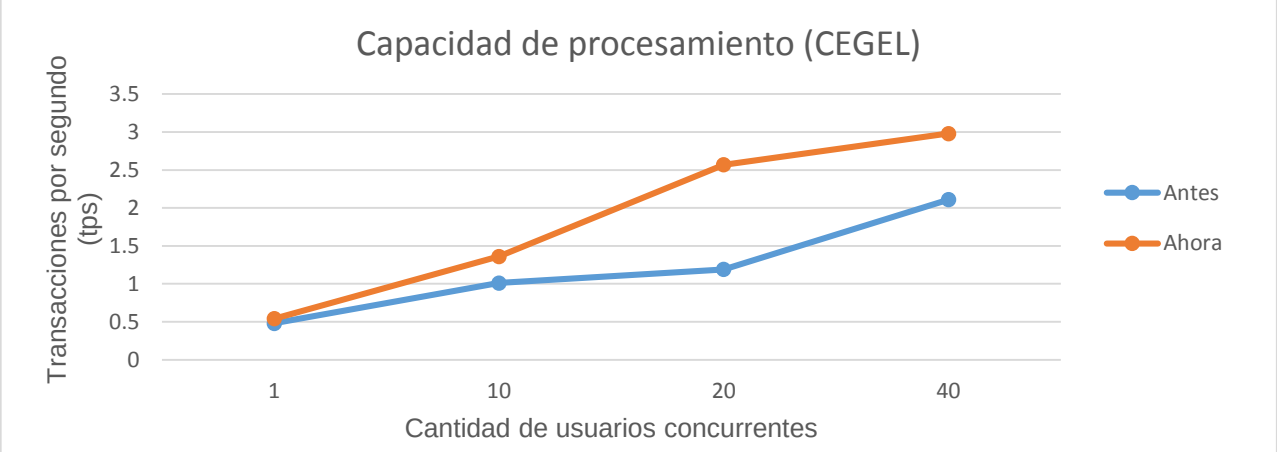

Figura 11: Capacidad de procesamiento promedio en el Centro CEGEL.

Las figuras 12, 13 y 14 ilustran el comportamiento del CPU durante la ejecución de las pruebas para cada Centro. Aunque se ve una mejora en la utilización del recurso, se observa que para una concurrencia de usuarios entre 20 y 40 nunca se logró cumplir con el criterio de aceptación de rendimiento; incluso para 10 usuarios concurrentes el promedio de uso siempre excedió el 50%.

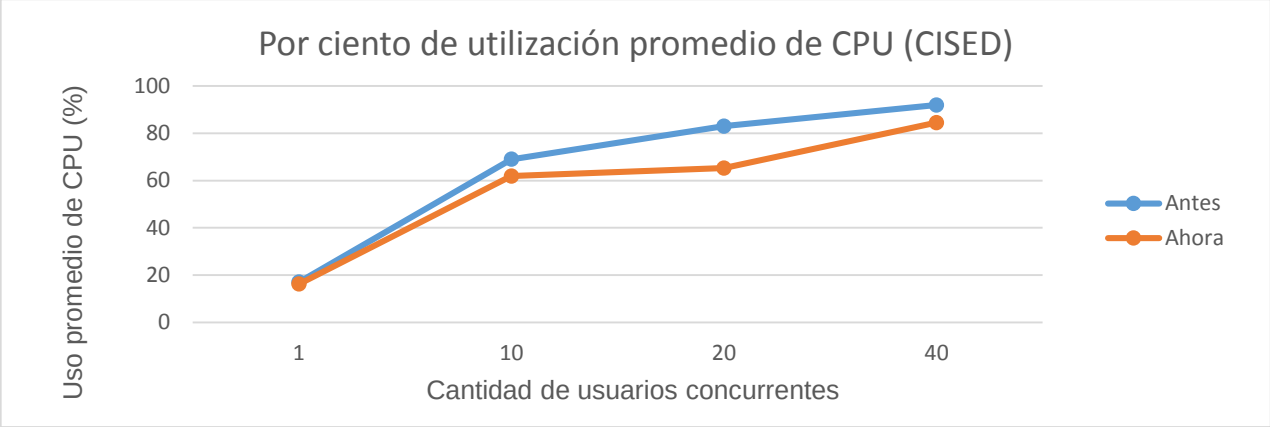

Figura 12: Por ciento de utilización promedio de CPU en el Centro CISED.

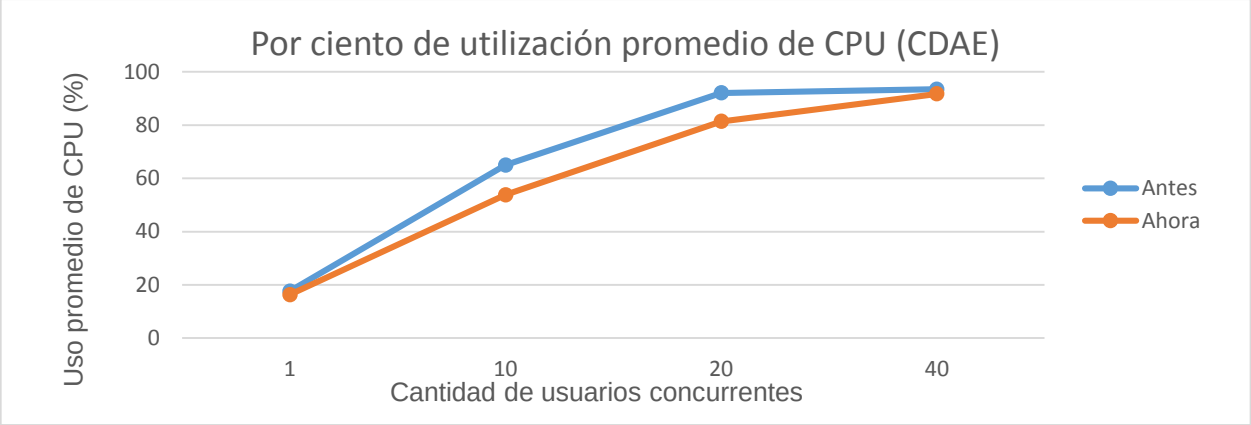

Figura 13: Por ciento de utilización promedio de CPU en el Centro CDAE.

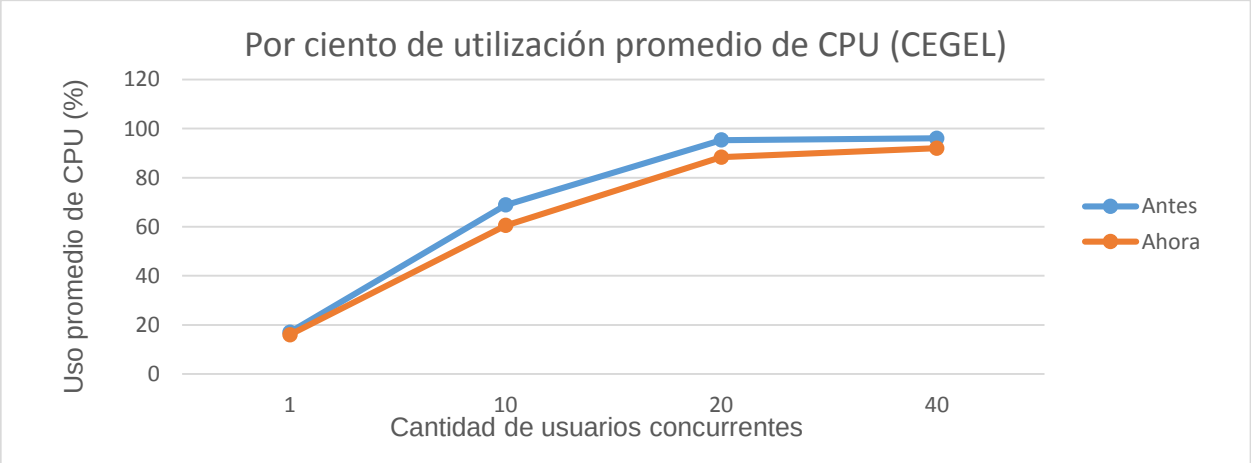

Figura 14: Por ciento de utilización promedio de CPU en el Centro CEGEL.

3.3.2 Caso de prueba Eliminar contenido:

Las figuras 15, 16 y 17 ilustran el comportamiento del tiempo de respuesta durante la ejecución de las pruebas para cada Centro. Se observa que en el Centro CISED se desproporciona para 40 usuarios debido a que el caso de prueba se ejecutó con errores inicialmente, no así después de ser ajustados los parámetros de la JVM. Aunque en los Centros CDAE y CEGEL se aprecia una mejora con respecto a los tiempos de respuesta, las pruebas se ejecutaron con errores para una concurrencia a partir de 20 usuarios.

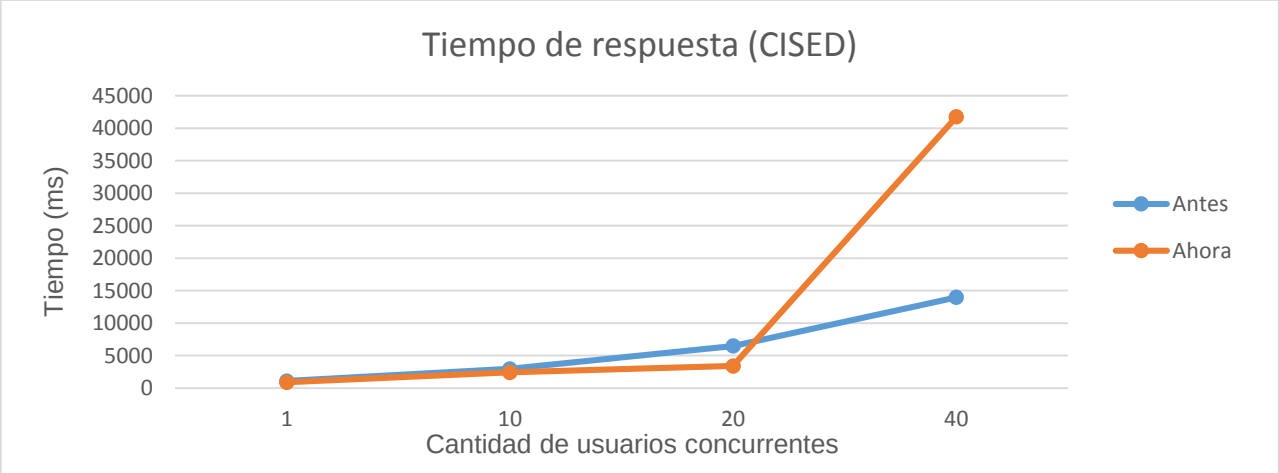

Figura 16: Tiempo de respuesta en el Centro CISED.

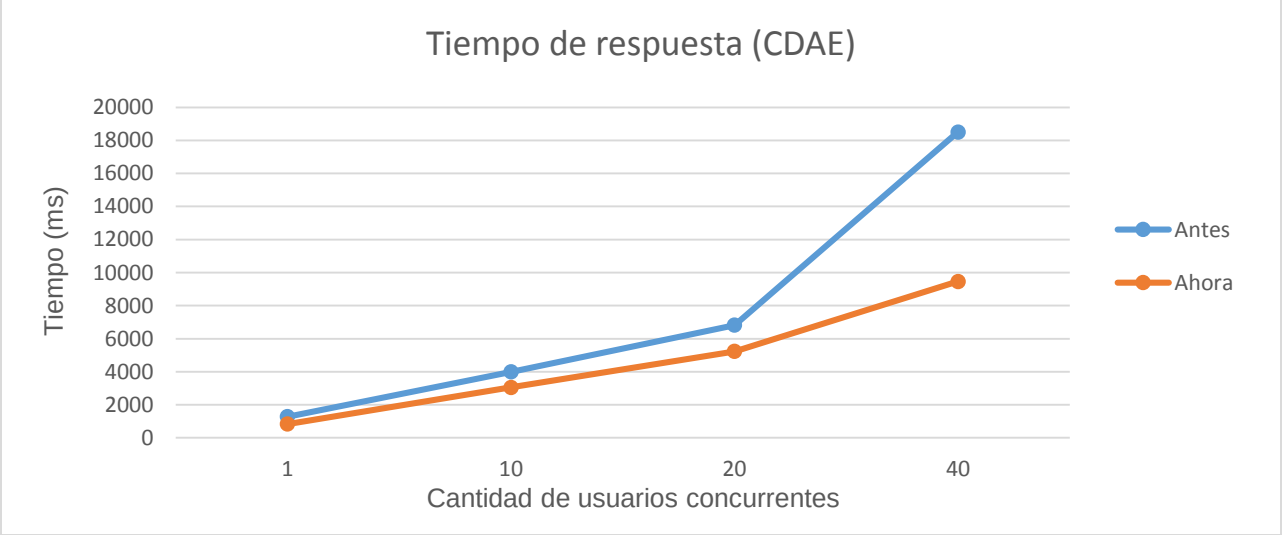

Figura 15: Tiempo de respuesta en el Centro CDAE.

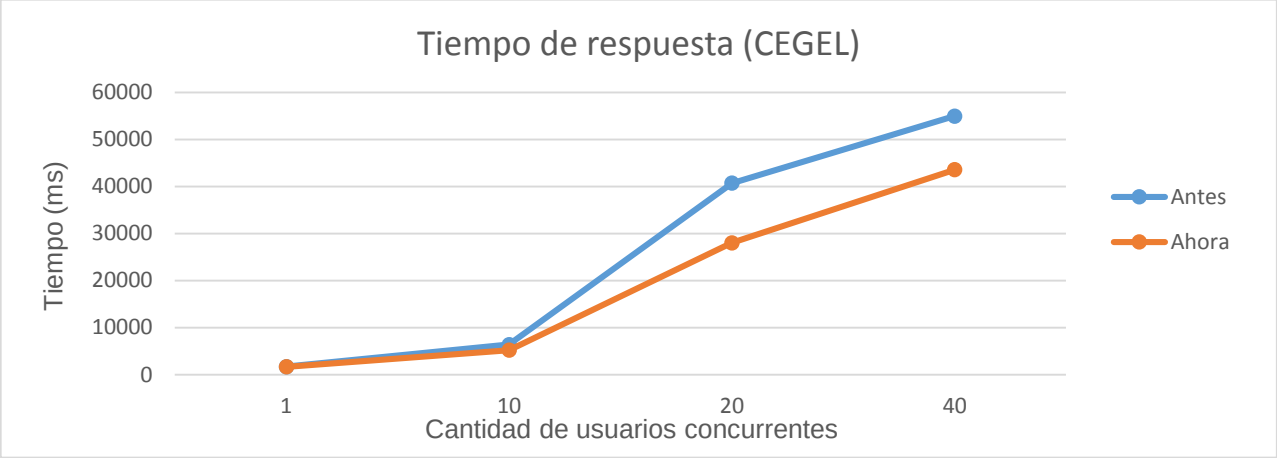

Figura 17: Tiempo de respuesta en el Centro CEGEL.

Las figuras 18, 19 y 20 ilustran el comportamiento de la capacidad de procesamiento del sistema para 1, 10, 20 y 40 usuarios concurrentes durante la ejecución de las pruebas para cada Centro. Aunque se muestra mejora en los resultados de las pruebas después de ser ajustados los parámetros de la JVM, se observa que dichos resultados son relativamente bajos.

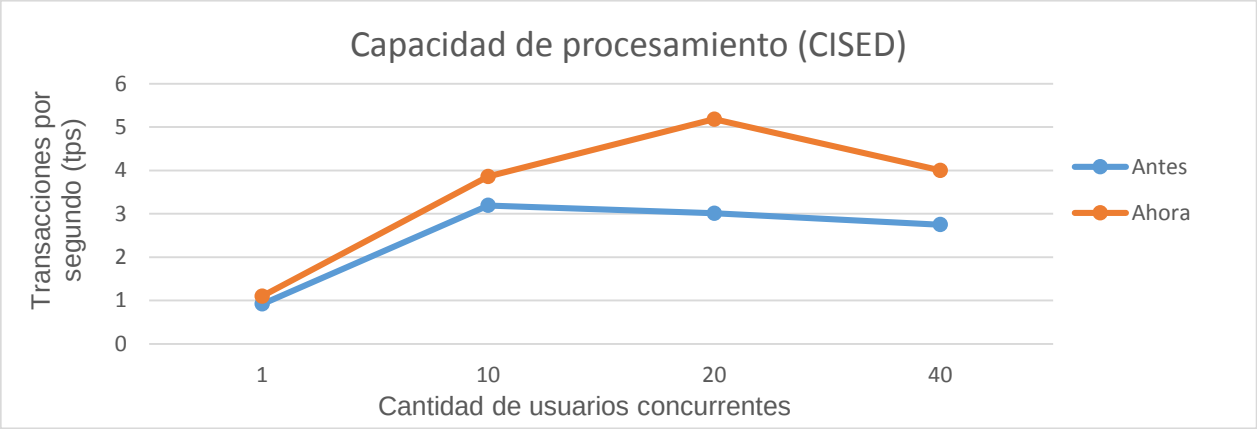

Figura 18: Capacidad de procesamiento en el Centro CISED.

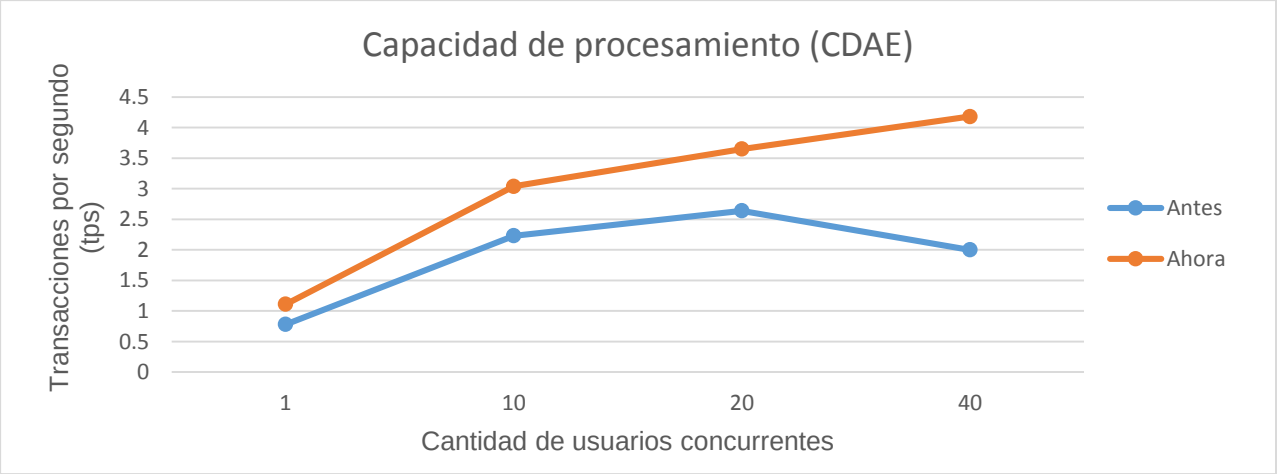

Figura 19: Capacidad de procesamiento en el Centro CDAE.

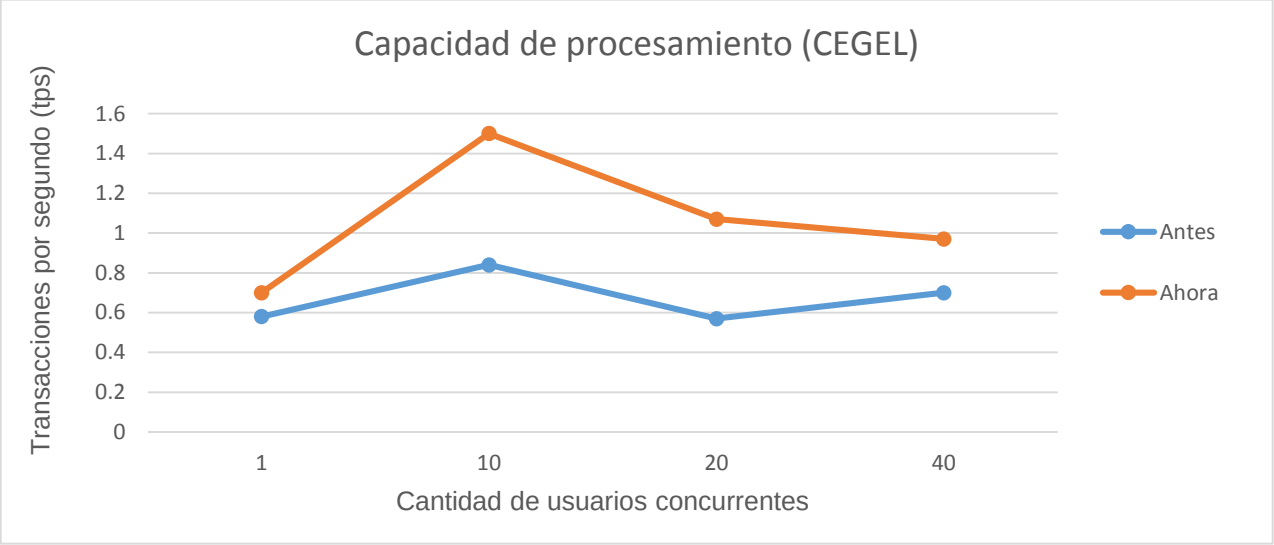

Figura 20: Capacidad de procesamiento en el Centro CEGEL.

Las figuras 21, 22 y 23 ilustran el comportamiento de la utilización promedio del CPU durante la ejecución de las pruebas para cada Centro. Se observa que para la ejecución del caso de prueba no se producen significativas mejoras en cuanto porciento de utilización promedio del CPU. Además, los porcientos de uso del CPU se mantienen elevados.

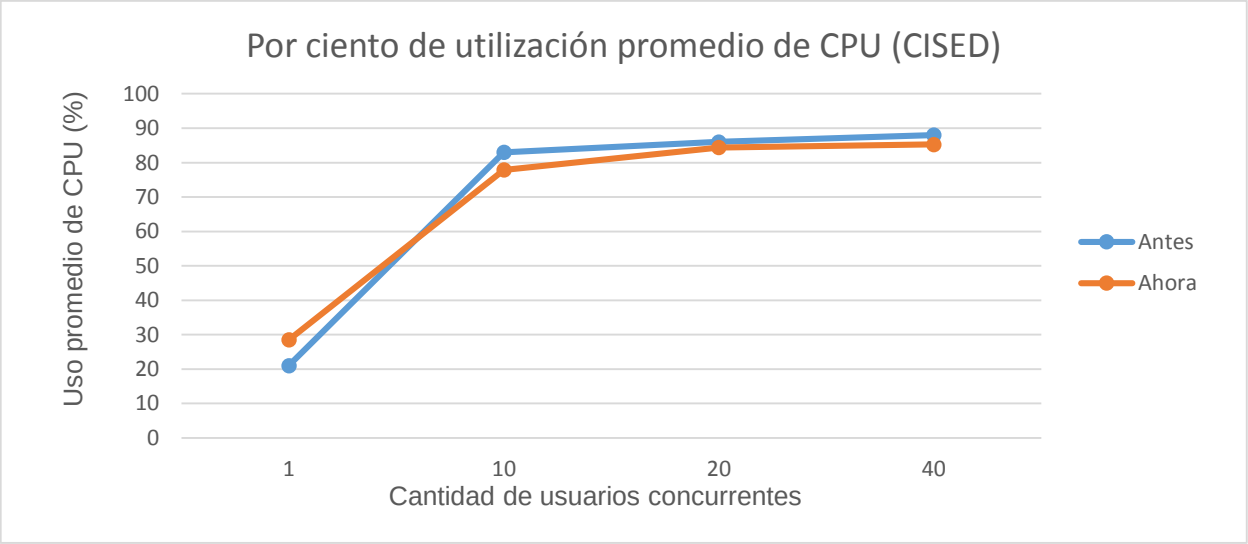

Figura 21: Por ciento de utilización promedio de CPU en el Centro CISED.

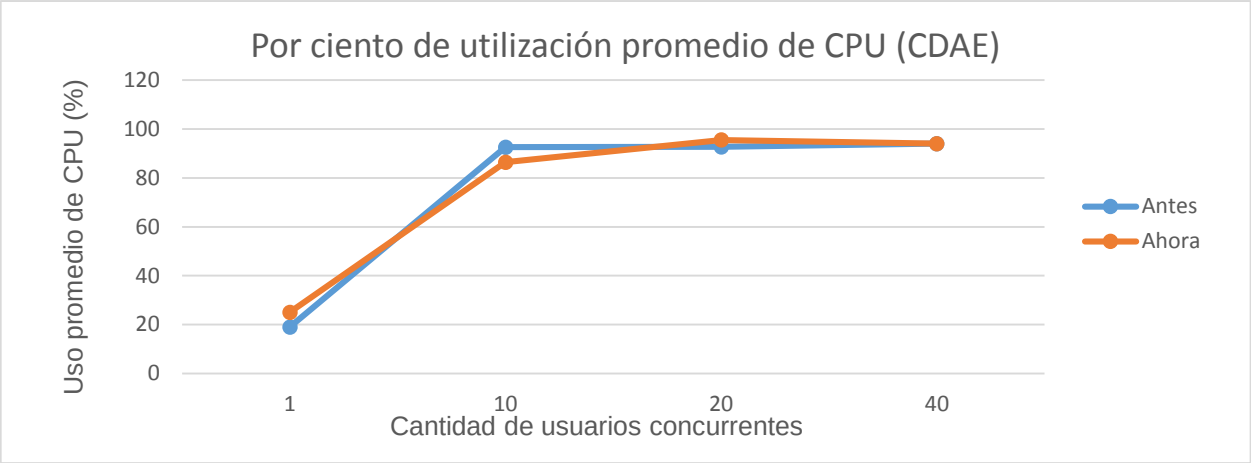

Figura 22: Por ciento de utilización promedio de CPU en el Centro CDAE.

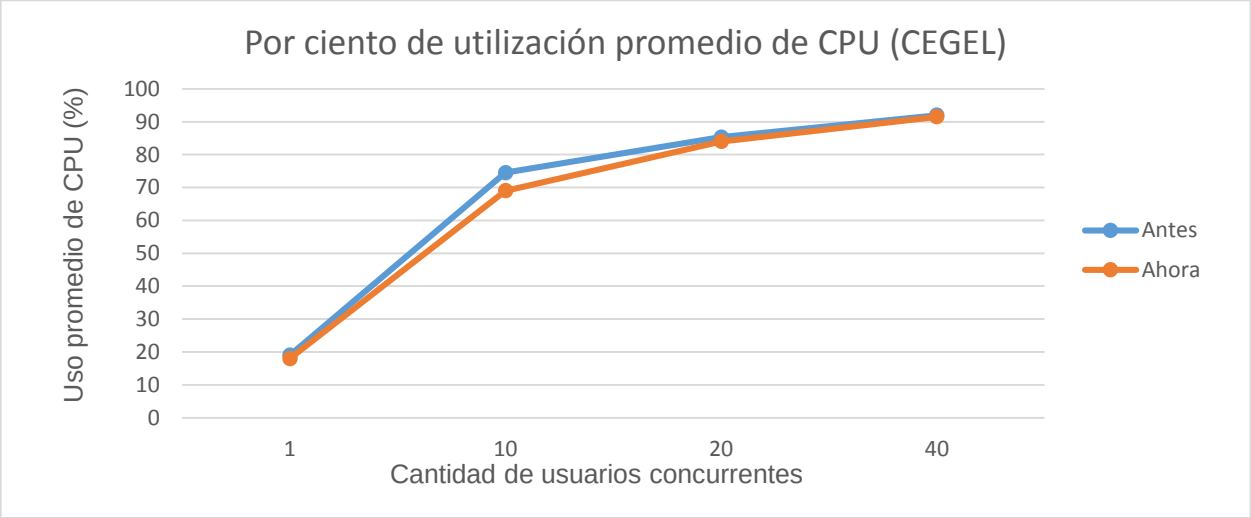

Figura 23: Por ciento de utilización promedio de CPU en el Centro CEGEL.

Los resultados obtenidos ilustran que el rendimiento del GDA eXcriba 2.0 desplegado en la Red de Centros de la Universidad de las Ciencias Informáticas no es el adecuado, pues se evidencia como promedio altos tiempos de respuesta a las peticiones realizadas al sistema, valores pequeños en la capacidad de procesamiento de la aplicación y porcientos relativamente altos en el uso del CPU, lo que indica, según Brad Micklea, ver epígrafe **1.1.2 Relación entre métricas de rendimiento**, que el sistema emplea la mayor parte de su tiempo en administrar el conflicto de los recursos en lugar de atender las peticiones realizadas.

3.4 Consideraciones para mejorar el rendimiento del sistema:

El estudio realizado sobre el desempeño permite identificar que para la optimización del rendimiento del sistema son necesarias un conjunto de acciones que proporcionen un cambio radical en el comportamiento que actualmente exhibe el GDA eXcriba 2.0 en la Red de Centros de la Universidad de las Ciencias Informáticas; pues a pesar que con las configuraciones realizadas en la JVM se observaron algunas

mejoras, estas métricas aún evidencian deficiencias en el desempeño del sistema en su entorno de producción.

Brad Micklea propone en (36) medir el tiempo de respuesta, la capacidad de procesamiento y la utilización de los recursos en varios niveles en los que se puede estructurar la arquitectura del sistema, teniendo como precepto que se basa en un modelo J2EE (en inglés *Java Enterprise Edition*). A continuación se ilustra:

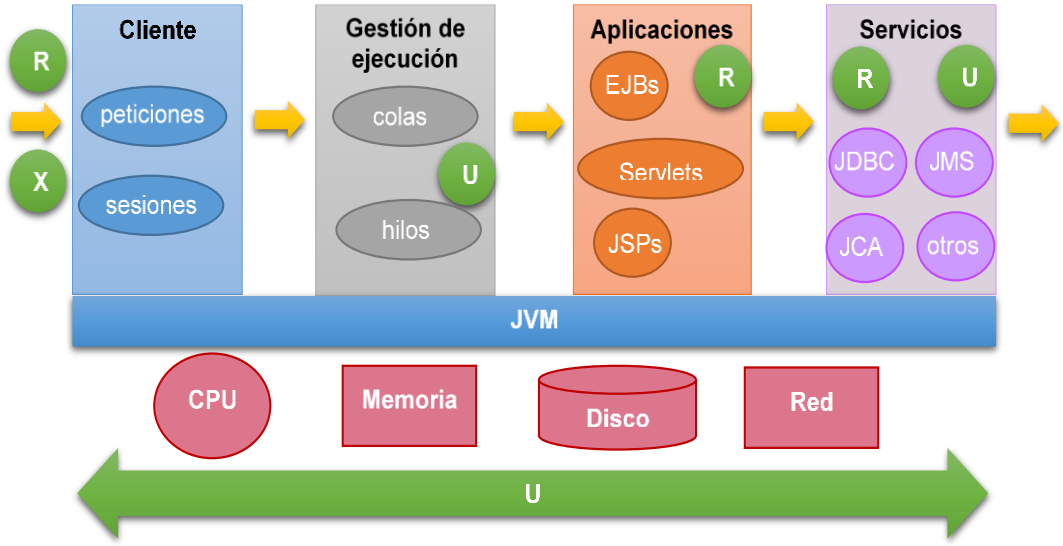

Figura 24: Modelo de un sistema J2EE con la aplicación de métricas de rendimiento.

R: Tiempo de respuesta.

X: Capacidad de procesamiento.

U: Utilización de los recursos.

El **Cliente** incluye las peticiones generadas por el usuario y las sesiones que pueden estar vinculadas a esas peticiones. La **Gestión de ejecución** es el proceso de asignar las peticiones en las colas antes de ser asignadas a los hilos de ejecución. El nivel de **Aplicaciones** incluye al código fuente y todos los componentes J2EE estándar. El nivel de **Servicios** incluye todo lo que conecta a la aplicación con el exterior. A su vez, todos estos elementos se ejecutan en la Máquina Virtual de Java, que por consiguiente utiliza los recursos del Sistema Operativo como procesador, memoria, disco y conexiones de red.

Por otra parte, teniendo en cuenta que el sistema utiliza los servicios que provee Alfresco 3.0 como repositorio de contenidos, se propone:

- Separar el proceso de indización del servidor de aplicaciones y del servidor de base de datos, escalar horizontalmente el sistema (37).
- Comprobar que la partición donde están los índices tenga el doble del tamaño total de los índices como espacio libre (38).
- Mantener los índices de Lucene en un disco local en vez de en un área de almacenamiento de red

(39).

- Verificar que el Sistema Operativo del servidor esté certificado oficialmente por la compañía Alfresco y que la Máquina Virtual de Java sea Sun JDK (en inglés *Java Development Kit*), preferiblemente de 64 bits (40).
- Utilizar la versión 6.0 del Tomcat que provee Apache en su sitio oficial.

Conclusiones parciales

- Las pruebas iniciales realizadas al sistema demostraron la existencia de deficiencias en el rendimiento del GDA eXcriba 2.0 en su entorno de producción, observándose que los resultados no cumplieron con los criterios de aceptación de rendimiento definidos para el tiempo promedio de respuesta y el por ciento de utilización promedio de CPU.
- La evaluación inicial de la capacidad de procesamiento del GDA eXcriba 2.0 y la observación del comportamiento de la memoria RAM dio como resultado que la capacidad del sistema para procesar las peticiones no excedió de 6 transacciones por segundo. Por otra parte, la utilización de la memoria RAM se incrementó a medida que fue aumentando la carga en el sistema, donde para los Centros CISED y CDAE alcanzaron valores próximos a la capacidad del recurso, mientras que para CEGEL no sobrepasó la mitad de la capacidad.
- Los resultados de las pruebas realizadas al GDA eXcriba 2.0, teniendo en cuenta el ajuste de parámetros de la Máquina Virtual de Java, demostraron que las configuraciones de estas opciones no constituyen un factor determinante que incide en el rendimiento del sistema en la Red de Centros de Desarrollo de la Universidad de las Ciencias Informáticas.

Conclusiones

- El análisis teórico del rendimiento como atributo de calidad permitió que la autora determinara en su investigación medir el desempeño del sistema teniendo en cuenta la capacidad de procesamiento, el tiempo de respuesta y la utilización de los recursos a partir de la ejecución de pruebas de rendimiento.
- La evaluación del rendimiento del GDA eXcriba 2.0 en la Red de Centros de la Universidad de las Ciencias Informáticas demostró la existencia de deficiencias en el desempeño del sistema en su entorno de producción: altos tiempos de respuesta a las peticiones realizadas al sistema, valores pequeños en la capacidad de procesamiento de la aplicación y porcientos relativamente altos en el uso del CPU.
- Las pruebas de rendimiento realizadas al sistema permitieron caracterizar su desempeño en el entorno de producción, posibilitando la definición de las métricas de rendimiento del Gestor de Documentos Administrativos eXcriba 2.0 en la Red de Centros de Desarrollo de la Universidad de las Ciencias Informáticas.

Recomendaciones

Se recomienda al equipo de desarrollo:

- Aplicar las pruebas de rendimiento como parte del ciclo de desarrollo de futuras versiones del GDA eXcriba.
- Realizar pruebas de rendimiento de capacidad para planear el futuro crecimiento de usuarios o de información en la base de datos y en el sistema de ficheros.
- Medir el comportamiento de los recursos del sistema teniendo en cuenta los recursos de red y disco.
- Analizar el impacto de los sistemas externos que coexisten en la misma virtualización del GDA eXcriba 2.0 en cada uno de los Centros.

Se recomienda al equipo que atiende los recursos en el centro de datos de la universidad:

- Distribuir equitativamente los recursos de *hardware* destinados para el funcionamiento del GDA eXcriba 2.0 en la Red de Centros de la universidad; debido a que se detectó mediante las pruebas de rendimiento, que en el Centro CEGEL solo se consume la mitad de la capacidad que se tiene destinada para la memoria RAM, mientras que en CISED y CDAE el consumo se manifiesta con escasa diferencia al límite de su capacidad.

Referencias Bibliográficas

1. **The Institute of Electrical and Electronics Engineers.** *IEEE Standard Glossary of Software Engineering Terminology Std 610.12-1990.* New York : Standards Coordinating Commitee of the Computer Society of the IEEE, 1990. 1-55937-067-X.
2. **Gorton, Ian.** Software Qualitty Attributes. *Essential Software Architecture.* 1ra. s.l. : Springer-Verlag, 2011, pág. 304.
3. **Meier, J.D, y otros.** Chapter 1 - Fundamentals of Engineering for Performance. *Improving .NET Application Performance and Scalability.* [En línea] Microsoft Corporation, Abril de 2004. [Citado el: 20 de Febrero de 20013.] http://msdn.microsoft.com/en-us/library/ff647781.aspx.
4. **Precise Software Solutions, Inc.** Solving J2EE Performance Problems. *WHITE PAPER Application Performance Management.* 2009.
5. **Barchini, Graciela.** Métodos I+D de la Informática. *Revista de Informática Educativa y Medios Audiovisuales.* 2005, Vol. II, 5, págs. 16-24.
6. **Microsoft Corporation.** Charpter 16 Quality Attributes. *Microsoft Application Architecture Guide.* [En línea] Octubre de 2009. [Citado el: 20 de Febrero de 2013.] http://msdn.microsoft.com/en-us/library/ee658094.aspx. 9780735627109.
7. **Zhu, Hui.** Quality-Attribute Auditing: The What, Why, and How. *Microsoft Developer Network.* [En línea] Mycrosoft, Mayo de 2007. [Citado el: 15 de Mayo de 2013.] http://msdn.microsoft.com/en-us/library/bb508961.aspx.
8. **Bass, Len, Clements, Paul y Kazman, Rick.** *Software Architecture in Practice.* s.l. : Addison-Wesley Publishing Co., 1998. 0-321-15495-9.
9. **Grady, R.B y Caswell, D.L.** *Software Metrics: Establishing a Company-Wide Program.* s.l. : Prentice-Hall, 1987.
10. **UNE-ISO.** ISO 9126 - Software Quality Model. *ISO 9126 - Software Quality Characteristics.* [En línea] [Citado el: 3 de Febrero de 2013.] http://www.sqa.net/iso9126.html.
11. **Barbacci, Mario, y otros.** *Quality Attributes. Technical Report CMU/SEI-95-TR-021 ESC-TR-95-021.* Pittsburgh : Software Engineering Institute, 1995.
12. **Jain, R.** *The Art of Computer Systems Performance Analysis.* New York : Wiley, 1991. 0471503363.
13. **Pressman, Roger.** Capítulo 20: Cómo probar aplicaciones web. *Ingeniería de Software: Un Enfoque Práctico.* 6ta. México : Mc Graw-Hill, 2005.
14. **Micklea, Brad.** Meassuring J2EE Application Performance in Production. [En línea] 2003. [Citado el: 25 de Febrero de 2013.] http://www.softwaresummit.com/2003/speakers/MickleaProductionPerformance.pdf.
15. **Choque Aspiazu, Guillermo.** Ingeniería de rendimiento del software. *Mente Errabunda.* [En línea] 20

de Agosto de 2010. [Citado el: 17 de Febrero de 2013.] http://menteerrabunda.blogspot.com/2010/08/ingenieria-de-rendimiento-del-software.html.

16. **Woodside, Murray, Franks, Greg y Petriu, Dorina C.** The Future of Software Performance Engineering. *2007 Future of Software Engineering.* Washington : IEEE Computer Society, 2007, págs. 171-187.
17. **Presman, Roger.** Capítulo 17: Técnicas de Prueba del Software. *Ingeniería de Software. Un Enfoque Práctico.* 5ta. s.l. : Mc Graw-Hill, 2002, Vol. I.
18. **Hetzel, William C.** *The Complete Guide to Software Testing.* 2da. 1988. pág. 280. 0894352423.
19. **Myers, Glenford J.** *The art of software testing.* New York : Wiley, 1979. 0471043281.
20. **Meier, J. D., y otros.** Chapter 1 – Fundamentals of Web Application Performance Testing. *Performance Testing Guiadance for Web Applications.* [En línea] Microsoft Corporation, Septiembre de 2007. [Citado el: 16 de Febrero de 2013.] http://msdn.microsoft.com/en-us/library/bb924356.aspx.
21. **Pressman, Roger.** Capítulo 13: Estrategias de pruebas del software. *Ingeniería del Software. Un Enfoque Práctico.* 6ta. México : Mc Graw-Hill, 2005.
22. **González, José U.** Pruebas de Rendimiento: Tipos y objetivos. *QA Técnico.* [En línea] Marzo de 2012. [Citado el: 16 de Febrero de 2013.] http://qatecnico.blogspot.com/2012/03/pruebas-de-rendimiento-tipos-y.html.
23. **Testhouse Consultores.** Pruebas de Rendimiento. *testhouse.* [En línea] 2012. [Citado el: 16 de Febrero de 2013.] http://www.es.testhouse.net/pruebas-de-rendimiento/.
24. **Meier, J. D., y otros.** Charpter 4 - Web Application Performance Testing Core Activities. *Performance Testing Guiadance for Web Applications.* [En línea] Microsoft Corporation, Septiembre de 2007. [Citado el: 16 de Febrero de 2013.] http://msdn.microsoft.com/en-us/library/bb924359.aspx.
25. —. Charpter 2 - Types of Performance Testing. *Performance Testing Guiadance for Web Applications.* [En línea] Microsoft Corporation, Septiembre de 2007. [Citado el: 16 de Febrero de 2013.] http://msdn.microsoft.com/en-us/library/bb924357.aspx.
26. **Díaz, Francisco J., y otros.** *Evaluación de herramientas Free/Open Source para pruebas de software.* [Document] Buenos Aires : Universidad de La Plata, 2011.
27. **Corey Goldberg.** Performance & Scalability Testing - Web Services. *Pylot- Web Performance Tool.* [En línea] 2009. [Citado el: 7 de Marzo de 2013.] http://www.pylot.org/.
28. **Apache.** *Apache JMeter Wiki.* [En línea] Apache. [Citado el: 2 de Abril de 2013.] http://wiki.apache.org/jmeter/.
29. **OpenSTA.** OpenSTA User Home for those Researching, Learning and Using OpenSTA. [En línea] 2013. [Citado el: 13 de Marzo de 2013.] http://opensta.org/.
30. **Aston, Philip; Fitzgerald, Calum.** The Grinder 3. [Online] Marzo 29, 2013. [Cited: Marzo 29, 2013.] http://grinder.sourceforge.net/g3/manual.html.
31. **Contributors Group, JMeter.** How many threads does JMeter support? *Jmeter Wiki.* [En línea] 27 de

Octubre de 2001. [Citado el: 2 de Abril de 2013.] http://wiki.apache.org/jmeter/HowManyThreads.

32. —. JMeter Distributed Testing Step-by-step. *Apache JMeter User's Manual.* [En línea] [Citado el: 5 de abril de 2013.] http://jmeter.apache.org/usermanual/jmeter_distributed_testing_step_by_step.pdf.
33. **Khanduja, Jaideep.** What is a Testing environment for software testing? *Quality Assurance and Project Management.* [En línea] 12 de 09 de 2008. [Citado el: 15 de 05 de 2013.] http://itknowledgeexchange.techtarget.com/quality-assurance/what-is-a-testing-environment-for-software-testing/.
34. **Meier, J. D., y otros.** Charpter 10 - Quantifying End-User Response Time Goals. *Performance Testing Guiadance for Web Applications.* [En línea] Microsoft Corporation, Septiembre de 2007. [Citado el: 20 de Marzo de 2013.] http://msdn.microsoft.com/en-us/library/bb924365.aspx.
35. **Collard, Ross.** Part 1: Developing de Test Strategy. *System Performance Testing. A Case Study.* New York : Collard & Company, 2005.
36. **Booch, Grady, Rumbaugh, James y Jacobson, Ivar.** Parte II: Los Flujos de Trabajo Fundamentales. *El Proceso Unificado de Desarrollo de Software.* s.l. : Addison Wesley, 2000.
37. **Moncrieff, Stuart.** How to write a performance test case. *MYLOADTEST.* [En línea] 14 de Febrero de 2005. [Citado el: 27 de Mayo de 2013.] http://www.myloadtest.com/how-to-write-a-performance-test-case/.
38. **Paul, Javin.** 10 points about Java Heap Space or Java Heap Memory. *Javarevisited.* [En línea] 6 de Mayo de 2011. [Citado el: 24 de Abril de 2013.] http://javarevisited.blogspot.co.uk/2011/05/java-heap-space-memory-size-jvm.html.
39. —. Difference between Stack and Heap memory in Java. *Javarevisited.* [En línea] 22 de Enero de 2013. [Citado el: 24 de Abril de 2013.] http://javarevisited.blogspot.com.au/2013/01/difference-between-stack-and-heap-java.html.
40. **Micklea, Brad.** Using Metrics to Optimize J2EE Application Performance in Production. *JDJ.* [En línea] 10 de Agosto de 2005. [Citado el: 4 de Mayo de 2013.] http://java.sys-con.com/node/117726.
41. **de la Fuente, Tony.** Arquitecturas en Alfresco. *: : blyx.com : : Blog : : Toni de la Fuente.* [En línea] 16 de Mayo de 2010. [Citado el: 17 de Marzo de 2013.] http://blyx.com/tag/open-source/.
42. **de la Fuente, Toni.** Algunos apuntes sobre Lucene en el mundo de Alfresco. *: : blyx.com : : Blog : : Toni de la Fuente : :.* [En línea] 7 de Diciembre de 2011. [Citado el: 24 de Mayo de 2013.] http://blyx.com/2011/12/07/algunos-apuntes-sobre-lucene-en-el-mundo-de-alfresco/.
43. **Alfresco Software, Inc.** Alfresco.org. *JVM Tuning.* [En línea] Alfresco Enterprise, 7 de Febrero de 2013. [Citado el: 24 de Abril de 2013.] http://wiki.alfresco.com/wiki/JVM_Tuning.
44. **Alfresco Software Inc.** Validating de architecture. *Alfresco Enterprise Documentation.* [En línea] [Citado el: 9 de Abril de 2013.] http://docs.alfresco.com/3.4/topic/com.alfresco.Enterprise_3_4_0.doc/tasks/configuration-checklist-

arch.html.

45. **Microsoft Corporation.** Charpter 16 Quality Attributes. *Microsoft Application Architecture Guide.* [En línea] Octubre de 2009. [Citado el: 20 de Febrero de 2013.] http://msdn.microsoft.com/en-us/library/ee658094.aspx. 9780735627109.
46. **Academia Española, Real.** *Diccionario de la Lengua Española - Vigésima segunda edición.* [En línea] 2001. [Citado el: 23 de 5 de 2013.] http://lema.rae.es/drae/.
47. **Pressman, Roger.** Capítulo 15: Métricas del Producto para el Software. *Ingeniería de Software: Un enfoque Práctico.* 6ta. México : Mc Graw-Hill, 2005.
48. *TM: a systematic methodology of software metrics.* **Ejiogu, Lem O.** 1, s.l. : SIGPLAN Notice, 1991, Vol. 26.
49. **Booch, G., Rumbaugh, J. y Jacobson, I.** *The Unified Software Development Process.* s.l. : Addison-Wesley, 1999. 0-201-57169-2.
50. **López, Eddie.** Introducción al Tema de la Virtualización. *Consulta un IT Pro.* [En línea] 24 de 09 de 2009. [Citado el: 10 de 5 de 2013.] http://www.consultaunitpro.com/tag/definicion-de-virtualizacion.
51. **IBM.** Parámetros de ajuste de la JVM de Sun HotSpot (Solaris y HP-UX). *IBM.* [En línea] IBM, 2013 de Enero de 2011. [Citado el: 12 de Abril de 2013.] http://pic.dhe.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=%2Fcom.ibm.websphere.express.doc%2Finfo%2Fexp%2Fae%2Frprf_hotspot_parms.html.
52. **Oracle.** Java HotSpot VM Options. *Oracle.* [En línea] Oracle. [Citado el: 12 de Abril de 2013.] http://www.oracle.com/technetwork/java/javase/tech/vmoptions-jsp-140102.html.
53. **DELL.** Red de área de almacenamiento (SAN). [En línea] [Citado el: 23 de 5 de 2013.] http://www1.la.dell.com/content/topics/topic.aspx/global/products/pvaul/topics/es/learn_storage_san?.

Bibliografía

1. **Academia Española, Real.** *Diccionario de la Lengua Española - Vigésima segunda edición.* [En línea] 2001. [Citado el: 23 de 5 de 2013.] http://lema.rae.es/drae/.
2. **Alfresco Software, Inc.** Alfresco.org. *JVM Tuning.* [En línea] Alfresco Enterprise, 7 de Febrero de 2013. [Citado el: 24 de Abril de 2013.] http://wiki.alfresco.com/wiki/JVM_Tuning.
3. **Alfresco Software Inc.** Validating de architecture. *Alfresco Enterprise Documentation.* [En línea] [Citado el: 9 de Abril de 2013.] http://docs.alfresco.com/3.4/topic/com.alfresco.Enterprise_3_4_0.doc/tasks/configuration-checklist-arch.html.
4. **Andalucía, Junta de.** Construcción de Planes de Prueba con JMeter. *Marco de Desarrollo de la Junta de Andalucía.* [En línea] [Citado el: 13 de Junio de 2013.] http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/391
5. **Barbacci, Mario, y otros.** *Quality Attributes. Technical Report CMU/SEI-95-TR-021 ESC-TR-95-021.* Pittsburgh : Software Engineering Institute, 1995.
6. **Barchini, Graciela.** Métodos I+D de la Informática. *Revista de Informática Educativa y Medios Audiovisuales.* 2005, Vol. II, 5, págs. 16-24.
7. **Bass, Len, Clements, Paul y Kazman, Rick.** *Software Architecture in Practice.* s.l. : Addison-Wesley Publishing Co., 1998. 0-321-15495-9.
8. **Booch, G., Rumbaugh, J. y Jacobson, I.** *The Unified Software Development Process.* s.l. : Addison-Wesley, 1999. 0-201-57169-2.
9. **Booch, Grady, Rumbaugh, James y Jacobson, Ivar.** Parte II: Los Flujos de Trabajo Fundamentales. *El Proceso Unificado de Desarrollo de Software.* s.l. : Addison Wesley, 2000.
10. **Choque Aspiazu, Guillermo.** Ingeniería de rendimiento del software. *Mente Errabunda.* [En línea] 20 de Agosto de 2010. [Citado el: 17 de Febrero de 2013.] http://menteerrabunda.blogspot.com/2010/08/ingenieria-de-rendimiento-del-software.html.
11. **Collard, Ross.** Part 1: Developing de Test Strategy. *System Performance Testing. A Case Study.* New York : Collard & Company, 2005.
12. **Contributors Group, JMeter.** How many threads does JMeter support? *Jmeter Wiki.* [En línea] 27 de Octubre de 2001. [Citado el: 2 de Abril de 2013.] http://wiki.apache.org/jmeter/HowManyThreads.
13. **Contributors Group, JMeter.** JMeter Distributed Testing Step-by-step. *Apache JMeter User's Manual.* [En línea] [Citado el: 5 de abril de 2013.] http://jmeter.apache.org/usermanual/jmeter_distributed_testing_step_by_step.pdf.
14. **Cordovez Mustelier, Sulamis.** Propuesta de Procedimiento y Herramienta para el desarrollo de Pruebas de Rendimiento de Carga y Estrés en el Grupo de Calidad FORTES. Trabajo de Diploma para

optar por el título de Ingeniero en Ciencias Informáticas, Universidad de las Ciencias Informáticas, La Habana, 2011.

15. **de la Fuente, Tony.** Arquitecturas en Alfresco. *: : blyx.com : : Blog : : Toni de la Fuente.* [En línea] 16 de Mayo de 2010. [Citado el: 17 de Marzo de 2013.] http://blyx.com/tag/open-source/.
16. **de la Fuente, Toni.** Algunos apuntes sobre Lucene en el mundo de Alfresco. *: : blyx.com : : Blog : : Toni de la Fuente.* [En línea] 7 de Diciembre de 2011. [Citado el: 24 de Mayo de 2013.] http://blyx.com/2011/12/07/algunos-apuntes-sobre-lucene-en-el-mundo-de-alfresco/.
17. **DELL.** Red de área de almacenamiento (SAN). [En línea] [Citado el: 23 de 5 de 2013.] http://www1.la.dell.com/content/topics/topic.aspx/global/products/pvaul/topics/es/learn_storage_san?
18. **Díaz, Francisco J., y otros.** *Evaluación de herramientas Free/Open Source para pruebas de software.* [Document] Buenos Aires : Universidad de La Plata, 2011.
19. **Ejiogu, Lem O.** *TM: a systematic methodology of software metrics.* 1, s.l. : SIGPLAN Notice, 1991, Vol. 26.
20. **González, José U.** Pruebas de Rendimiento: Tipos y objetivos. *QA Técnico.* [En línea] Marzo de 2012. [Citado el: 16 de Febrero de 2013.] http://qatecnico.blogspot.com/2012/03/pruebas-de-rendimiento-tipos-y.html.
21. **González, Fernando.** Avisos de Alfresco sobre saturación de la Cache (EHCache / Hibernate). *El Blog de FEGOR.* [En línea] 18 de Enero de 2011. [Citado el 24 de Mayo de 2013] http://www.fegor.com/2011/01/avisos-de-alfresco-sobre-saturacion-de.html
22. **González, Fernando.** Algo de tuning con grandes cantidades de documentos (I). *El Blog de FEGOR.* [En línea] 16 de Marzo de 2011. [Citado el 2 de Abril de 2013] http://www.fegor.com/2011/03/algo-de-tuning-con-grandes-cantidades.html
23. **González, Fernando.** Algo de tuning con grandes cantidades de documentos (II). *El Blog de FEGOR.* [En línea] 21 de Enero de 2011. [Citado el 2 de Abril de 2013] http://www.fegor.com/2011/03/algo-de-tuning-con-grandes-cantidades_21.html
24. **González, José U.** Pruebas de Rendimiento: Tipos y objetivos. *QA Técnico.* [En línea] Marzo de 2012. [Citado el: 16 de Febrero de 2013.] http://qatecnico.blogspot.com/2012/03/pruebas-de-rendimiento-tipos-y.html.
25. **Gorton, Ian.** Software Qualitty Attributes. *Essential Software Architecture.* 1ra. s.l. : Springer-Verlag, 2011, pág. 304.
26. **Grady, R.B y Caswell, D.L.** *Software Metrics: Establishing a Company-Wide Program.* s.l. : Prentice-Hall, 1987.
27. **Hernández Sampieri, Roberto, y otros.** Metodología de la Investigación. México, McGraw-Hill, 1998. 502 p. 970-10-1899-0.

28. **Hetzel, William C.** *The Complete Guide to Software Testing.* 2da. 1988. pág. 280. 0894352423.
29. **IBM.** Parámetros de ajuste de la JVM de Sun HotSpot (Solaris y HP-UX). *IBM.* [En línea] IBM, 2013 de Enero de 2011. [Citado el: 12 de Abril de 2013.] http://pic.dhe.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=%2Fcom.ibm.websphere.express.doc%2Finfo%2Fexp%2Fae%2Frprf_hotspot_parms.html.
30. **Jain, R.** *The Art of Computer Systems Performance Analysis.* New York : Wiley, 1991. 0471503363.
31. **Khanduja, Jaideep.** What is a Testing environment for software testing? *Quality Assurance and Project Management.* [En línea] 12 de 09 de 2008. [Citado el: 15 de 05 de 2013.] http://itknowledgeexchange.techtarget.com/quality-assurance/what-is-a-testing-environment-for-software-testing/.
32. **López, Eddie.** Introducción al Tema de la Virtualización. *Consulta un IT Pro.* [En línea] 24 de 09 de 2009. [Citado el: 10 de 5 de 2013.] http://www.consultaunitpro.com/tag/definicion-de-virtualizacion.
33. **Meier, J.D, y otros.** Chapter 1 - Fundamentals of Engineering for Performance. *Improving .NET Application Performance and Scalability.* [En línea] Microsoft Corporation, Abril de 2004. [Citado el: 20 de Febrero de 20013.] http://msdn.microsoft.com/en-us/library/ff647781.aspx.
34. **Meier, J. D., y otros.** Chapter 1 – Fundamentals of Web Application Performance Testing. *Performance Testing Guiadance for Web Applications.* [En línea] Microsoft Corporation, Septiembre de 2007. [Citado el: 16 de Febrero de 2013.] http://msdn.microsoft.com/en-us/library/bb924356.aspx.
35. **Meier, J. D., y otros.** Charpter 2 - Types of Performance Testing. *Performance Testing Guiadance for Web Applications.* [En línea] Microsoft Corporation, Septiembre de 2007. [Citado el: 16 de Febrero de 2013.] http://msdn.microsoft.com/en-us/library/bb924357.aspx.
36. **Meier, J. D., y otros.** Charpter 4 - Web Application Performance Testing Core Activities. *Performance Testing Guiadance for Web Applications.* [En línea] Microsoft Corporation, Septiembre de 2007. [Citado el: 16 de Febrero de 2013.] http://msdn.microsoft.com/en-us/library/bb924359.aspx.
37. **Meier, J. D., y otros.** Charpter 10 - Quantifying End-User Response Time Goals. *Performance Testing Guiadance for Web Applications.* [En línea] Microsoft Corporation, Septiembre de 2007. [Citado el: 20 de Marzo de 2013.] http://msdn.microsoft.com/en-us/library/bb924365.aspx.
38. **Myers, Glenford J.** *The art of software testing.* New York : Wiley, 1979. 0471043281.
39. **Micklea, Brad.** Meassuring J2EE Application Performance in Production. [En línea] 2003. [Citado el: 25 de Febrero de 2013.] http://www.softwaresummit.com/2003/speakers/MickleaProductionPerformance.pdf.
40. **Micklea, Brad.** Using Metrics to Optimize J2EE Application Performance in Production. *JDJ.* [En línea] 10 de Agosto de 2005. [Citado el: 4 de Mayo de 2013.] http://java.sys-con.com/node/117726.
41. **Microsoft Corporation.** Charpter 16 Quality Attributes. *Microsoft Application Architecture Guide.* [En

línea] Octubre de 2009. [Citado el: 20 de Febrero de 2013.] http://msdn.microsoft.com/en-us/library/ee658094.aspx. 9780735627109.

42. **Moncrieff, Stuart.** How to write a performance test case. *MYLOADTEST.* [En línea] 14 de Febrero de 2005. [Citado el: 27 de Mayo de 2013.] http://www.myloadtest.com/how-to-write-a-performance-test-case/.
43. **Oracle.** Java HotSpot VM Options. *Oracle.* [En línea] Oracle. [Citado el: 12 de Abril de 2013.] http://www.oracle.com/technetwork/java/javase/tech/vmoptions-jsp-140102.html.
44. **Paul, Javin.** 10 points about Java Heap Space or Java Heap Memory. *Javarevisited.* [En línea] 6 de Mayo de 2011. [Citado el: 24 de Abril de 2013.] http://javarevisited.blogspot.co.uk/2011/05/java-heap-space-memory-size-jvm.html.
45. **Paul, Javin.** Difference between Stack and Heap memory in Java. *Javarevisited.* [En línea] 22 de Enero de 2013. [Citado el: 24 de Abril de 2013.] http://javarevisited.blogspot.com.au/2013/01/difference-between-stack-and-heap-java.html.
46. **Precise Software Solutions, Inc.** Solving J2EE Performance Problems. *WHITE PAPER Application Performance Management.* 2009.
47. **Pressman, Roger.** Capítulo 13: Estrategias de pruebas del software. *Ingeniería del Software. Un Enfoque Práctico.* 6ta. México : Mc Graw-Hill, 2005.
48. **Pressman, Roger.** Capítulo 15: Métricas del Producto para el Software. *Ingeniería de Software: Un enfoque Práctico.* 6ta. México : Mc Graw-Hill, 2005.
49. **Presman, Roger.** Capítulo 17: Técnicas de Prueba del Software. *Ingeniería de Software. Un Enfoque Práctico.* 5ta. s.l. : Mc Graw-Hill, 2002, Vol. I.
50. **Pressman, Roger.** Capítulo 20: Cómo probar aplicaciones web. *Ingeniería de Software: Un Enfoque Práctico.* 6ta. México : Mc Graw-Hill, 2005.
51. **The Institute of Electrical and Electronics Engineers.** *IEEE Standard Glossary of Software Engineering Terminology Std 610.12-1990.* New York : Standards Coordinating Commitee of the Computer Society of the IEEE, 1990. 1-55937-067-X.
52. **Testhouse Consultores.** Pruebas de Rendimiento. *testhouse.* [En línea] 2012. [Citado el: 16 de Febrero de 2013.] http://www.es.testhouse.net/pruebas-de-rendimiento/.
53. **UNE-ISO.** ISO 9126 - Software Quality Model. *ISO 9126 - Software Quality Characteristics.* [En línea] [Citado el: 3 de Febrero de 2013.] http://www.sqa.net/iso9126.html.
54. **Woodside, Murray, Franks, Greg y Petriu, Dorina C.** The Future of Software Performance Engineering. *2007 Future of Software Engineering.* Washington : IEEE Computer Society, 2007, págs. 171-187.
55. **Zhu, Hui.** Quality-Attribute Auditing: The What, Why, and How. *Microsoft Developer Network.* [En línea] Mycrosoft, Mayo de 2007. [Citado el: 15 de Mayo de 2013.] http://msdn.microsoft.com/en-us/library/bb508961.aspx.

Glosario de Términos

Carga: Cantidad de usuarios concurrentes en un sistema.

Escalabilidad: se define como la capacidad de un sistema para manejar cualquier aumento en la carga de trabajo sin afectar el rendimiento del sistema, o bien, la capacidad de ampliarse fácilmente (41).

JMS: El Servicio de Mensajes Java (en inglés, *Java Message Service*) creado para el uso de colas de mensajes. Permite a los componentes de aplicaciones basados en la plataforma Java2 crear, enviar, recibir y leer mensajes.

Kit: Conjunto de productos y utensilios suficientes para conseguir un determinado fin, que se comercializan como una unidad (42).

Medición: Acto de determinar una medida (43).

Medida: Indicación cuantitativa de la extensión, la cantidad, la dimensión, la capacidad o el tamaño de algún atributo de un producto o proceso (43).

Métrica: Medida cuantitativa del grado en el que un sistema, componente o proceso posee un atributo determinado (1).

Métrica de calidad: Una medida cuantitativa del grado en el que un elemento posee un atributo de calidad determinado (1).

Métricas de rendimiento: Miden la conducta de módulos y sistemas de un *software*, bajo la supervisión del Sistema Operativo o el *hardware* (44).

Requisito o requerimiento: Condición o capacidad que debe cumplir o poseer un sistema o un componente de un sistema para satisfacer un contrato, estándar, especificación u otros documentos formales requeridos (1).

Requisito funcional: Un requerimiento que especifica una función que un sistema o componente de un sistema debe ser capaz de realizar (1).

Requisito no funcional: Especifica propiedades del sistema, tales como restricciones de implementación y de ambiente, rendimiento, dependencias de plataforma, mantenimiento, capacidad de extensión y confiabilidad. Un requerimiento que especifica restricciones físicas sobre un requerimiento funcional (45).

Virtualización: Es una tecnología que permite instalar y configurar múltiples computadoras y/o servidores completamente independientes, conocidas como máquinas virtuales, en inglés *virtual machine*, en un único recurso físico como computadora, servidor. Cada máquina virtual trabaja de manera totalmente independiente aunque comparten todas los recursos de un mismo *hardware* (46).

Anexos

Anexo 1: Tipos de componentes del plan de prueba en JMeter

Tabla 24: Descripción de los elementos del plan de prueba en JMeter.

Elemento	Descripción
Testplan	Representa la raíz del árbol. En todo el plan de prueba existe uno y sólo un componente de este tipo
Thread Group	Representa un grupo de usuarios. En JMeter cada *thread* es un usuario virtual. Este tipo de componente permite representar grupos de usuarios. Cada grupo de usuarios del plan de prueba representa un rol o perfil. Todos los *threads* de un mismo *thread group* realizan la misma secuencia de acciones, representada por los *samplers* que agrupa el *thread group*.
Controllers: • *Sampler* • *Logic Controler*	Son los únicos componentes del plan de prueba que ejecutan acciones: los *samplers* realizan peticiones contra la aplicación, y los *logic controlers* establecen el orden en que se ejecutan los *samplers* que agrupan. El resto de componentes definen la forma en que se ejecutan los *samplers*, pero no varían sustancialmente su comportamiento.
Config Element	Establecen propiedades de configuración que se aplican a los *samplers* a los que afectan.
Assertion	Comprueban condiciones aplicadas a las peticiones que realizan, los *samplers* a los que afectan, a la aplicación.
Listener	Recopilan datos de las peticiones que realizan los *samplers* a los que afectan.
Timer	Añaden tiempo extra a la ejecución de las peticiones que realizan a la aplicación los *samplers* a los que afectan
Pre-Processor element	Realizan acciones o establecen configuraciones, previa a la ejecución de los *samplers* a los que afectan.
Post-Processor element	Realizan acciones o establecen configuraciones posteriormente a la ejecución de los *samplers* a los que afectan.

Anexo 2: Descripción de las opciones de la Máquina Virtual de Java

A continuación se relacionan los parámetros de la JVM que se han enunciado con anterioridad (47), (48):

Tabla 25: Descripción de las opciones de configuración de la JVM.

Parámetro	Descripción
`-server`	Modo para ejecutar la JVM. Se utiliza por defecto para arquitecturas 64 bit. Requiere más tiempo para iniciar la JVM pero incrementa el rendimiento de los procesos durante la ejecución.
`-Xcomp`	Utilizado para compilar con antelación todo el código con la máxima optimización. Su utilización puede incrementar el tiempo de inicio de la aplicación.
`-Xbatch`	Detiene los procesos de compilación que se ejecutan en segundo plano. Su utilización puede incrementar el tiempo de inicio de la aplicación.
`-Xss`	Determina el tamaño máximo de la pila para los hilos de Java. Si el valor asignado es demasiado pequeño pueden ocurrir excepciones como *class java.lang.StackOverflowError*.
`-Xms`	Para especificar el tamaño de almacenamiento dinámico inicial para la JVM.
`-Xmx`	Para especificar el tamaño de almacenamiento dinámico máximo para la JVM.
`-XX:MaxPermSize`	Especifica el tamaño de la Generación Permanente.
`-XX:NewSize`	Se debe configurar si utiliza el recolector de marca y barrido de pausa baja simultánea. El tamaño de generación joven actual va a ser mayor o igual que el tamaño de generación joven inicial o mínimo, como se especifica en el parámetro `-XX:NewSize`.
`-XX:NewRatio`	Este valor suele ofrecer un buen rendimiento en la recogida de basura generacional. El valor predeterminado de 2 significa que la región de tenencia es el doble del tamaño de la región de la generación joven, es decir, que la generación joven es un tercio de la totalidad del almacenamiento dinámico Java.
`-XX:+UseConcMarkSweepGC`	Reduce la fragmentación del *Heap*. Este parámetro habilita al recolector de marca y barrido de pausa baja simultánea. Esta modalidad reconfigura la recogida de basura generacional de forma tal que se minimicen las pausas invasivas introducidas al tener que recopilar el contenido de la generación joven.

`-XX:+CMSIncrementalMode`	
`-XX:CMSInitiatingOccupanceyFraction`	Se debe configurar si se utiliza el recolector de marca y barrido de pausa baja simultánea. Permite desencadenar la recogida de basura para recuperar espacio suficiente antes del inicio de la asignación. Comprueba el porcentaje de utilización del almacenamiento dinámico Java para desencadenar la recogida de basura. Su valor predeterminado es de un 70%.

Anexo 3: Modelo para el diseño de casos de prueba de rendimiento

Tabla 26: Modelo para el diseño de casos de prueba de rendimiento.

Caso de prueba:	*[Nombre del caso de prueba]*
Descripción:	*[Breve descripción de lo que permite el caso de prueba]*
Requisitos:	*[Requerimientos que se deben cumplir para que se ejecute el caso de prueba]*

No.	Descripción	Resultado esperado	Nombre	Tiempo
[Número de la transacción]	*[Descripción de la transacción que se realiza]*	*[Respuesta que se espera al ejecutar la transacción]*	*[Nombre asignado a la transacción]*	*[Tiempo estimado en segundos que pudiera tardar el usuario para ejecutar la transacción]*

Anexo 4: Utilización de los recursos en el entorno de prueba

Caso de prueba Adicionar contenido

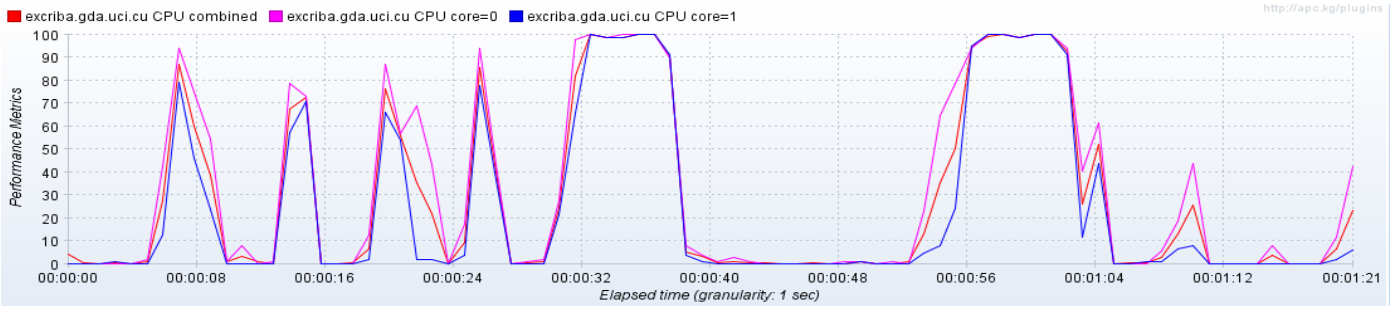

Figura 25: Utilización del CPU un usuario caso de prueba Adicionar contenido.

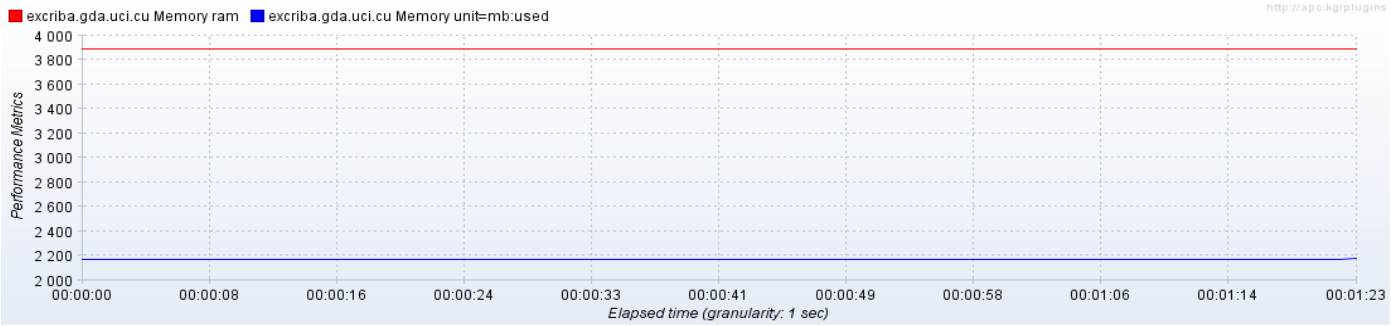

Figura 26: Utilización de la memoria un usuario caso de prueba Adicionar contenido.

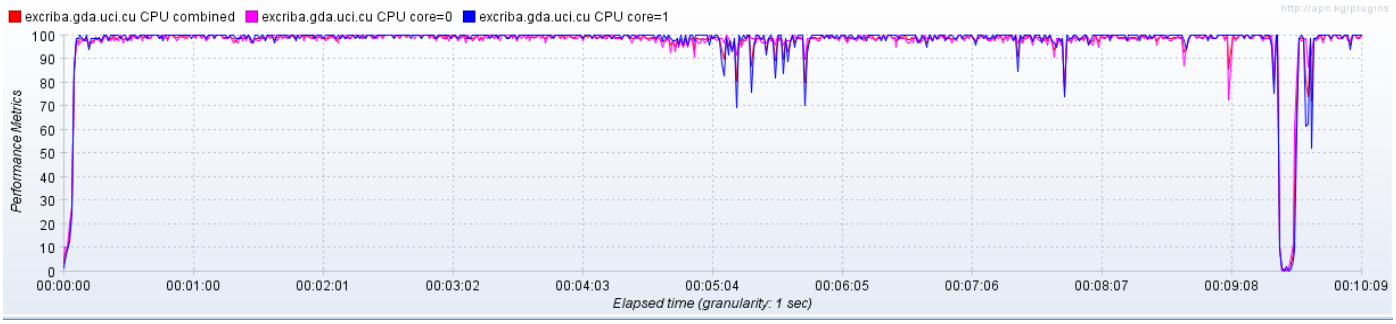

Figura 27: Utilización del CPU 40 usuarios caso de prueba Adicionar contenido.

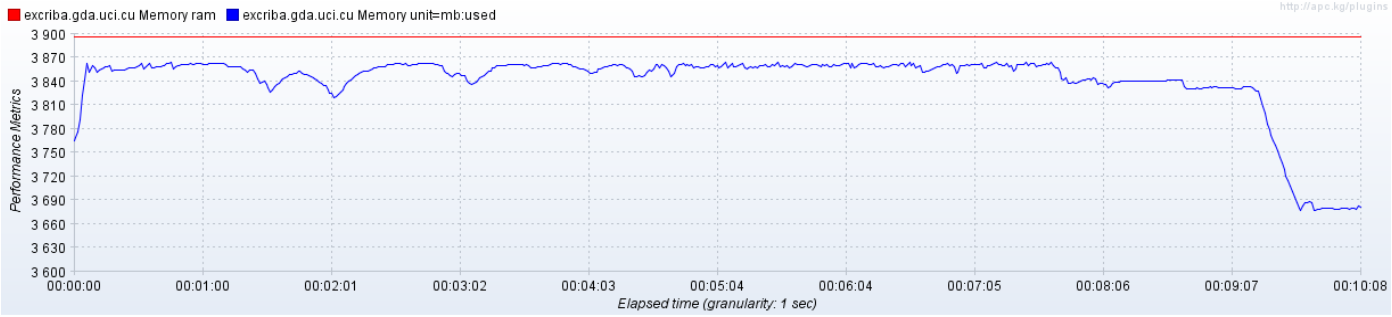

Figura 28: Utilización de la memoria 40 usuarios caso de prueba Adicionar contenido.

Caso de prueba Eliminar contenido

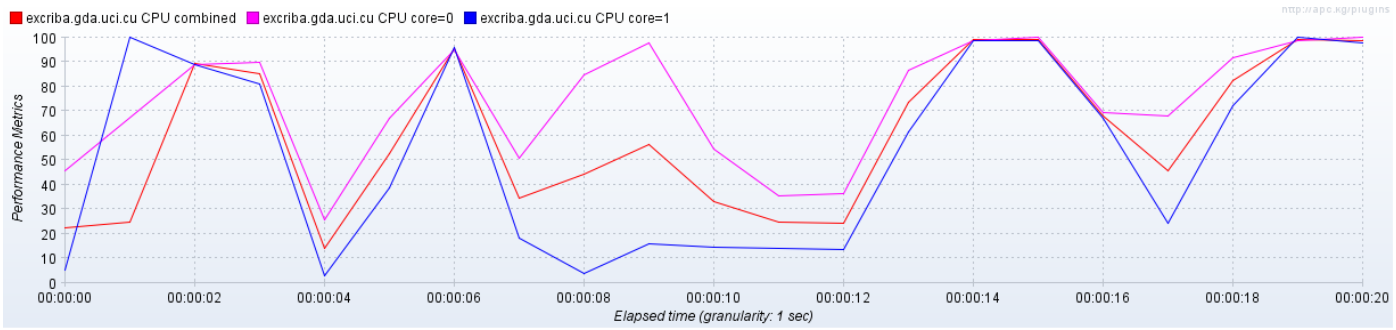

Figura 29: Utilización del CPU un usuario caso de prueba Eliminar contenido.

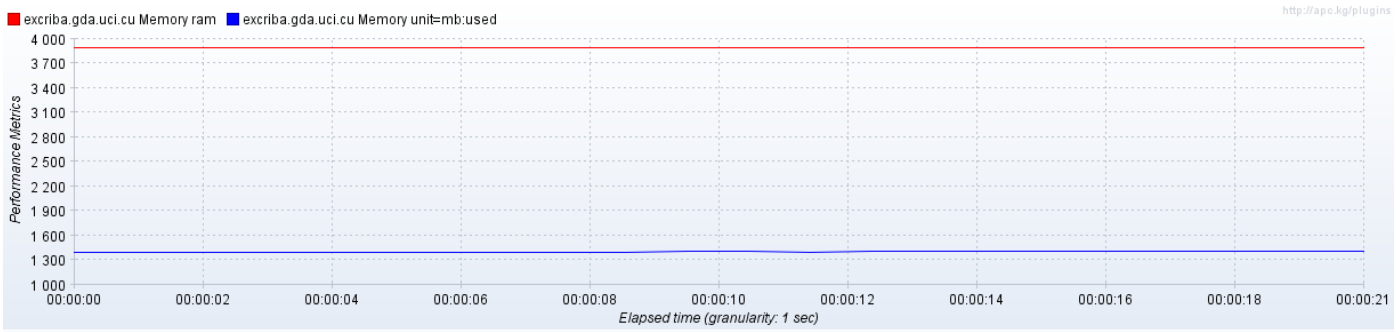

Figura 30: Utilización de la memoria un usuario caso de prueba Eliminar contenido.

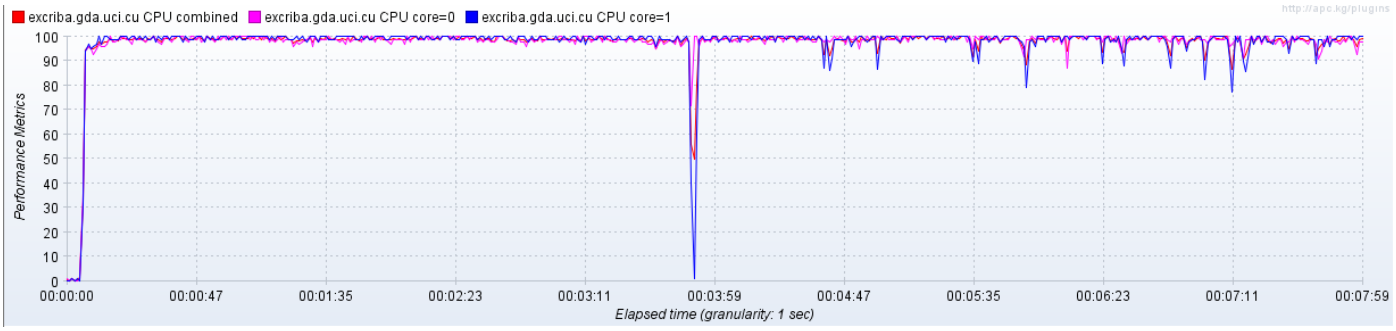

Figura 31: Utilización del CPU 40 usuarios caso de prueba Eliminar contenido.

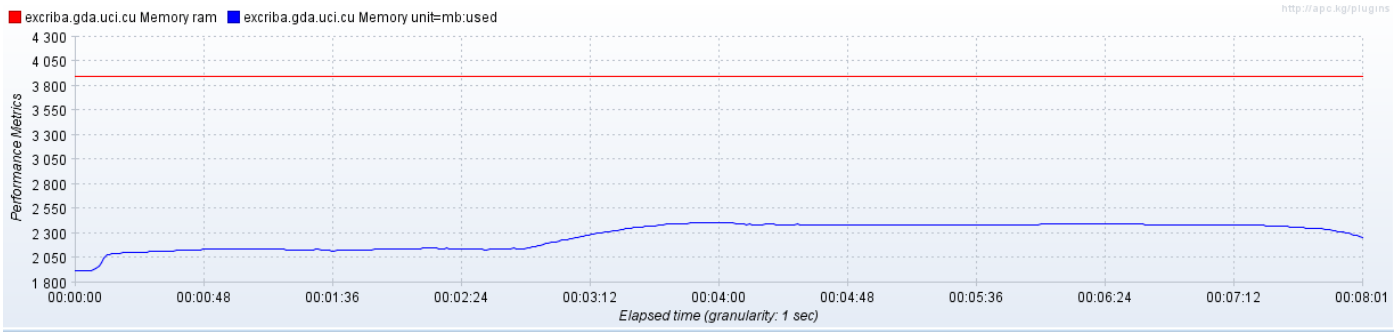

Figura 32: Utilización de la memoria 40 usuarios caso de prueba Eliminar contenido.

Anexo 6: Utilización de los recursos en el Centro CISED

Caso de prueba Adicionar contenido

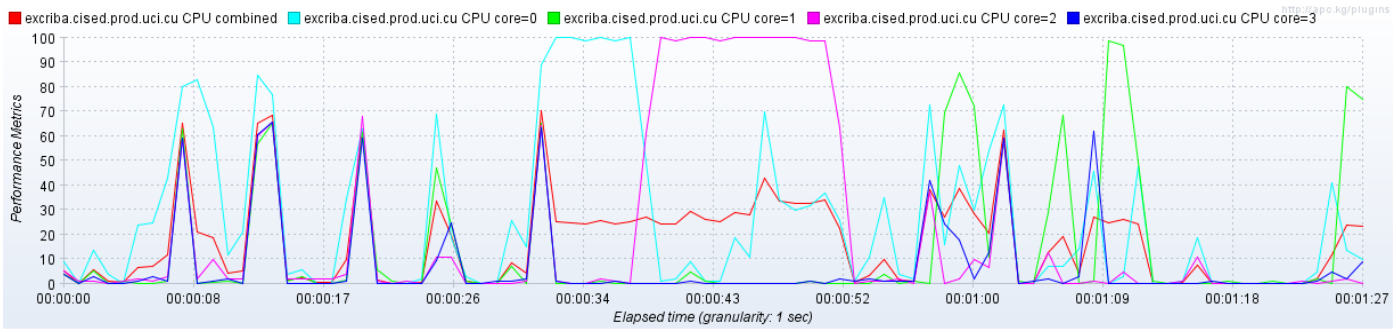

Figura 33: Utilización del CPU un usuario caso de prueba Adicionar contenido.

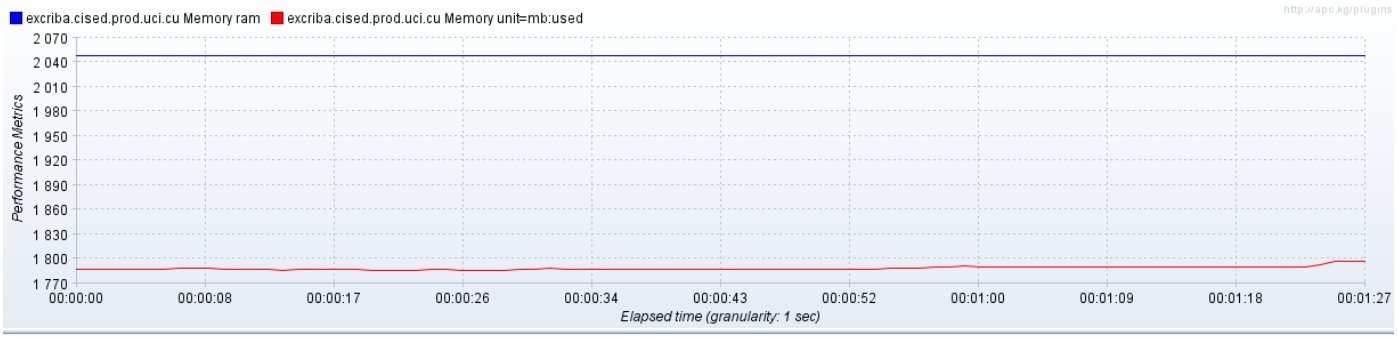

Figura 34: Utilización de la memoria un usuario caso de prueba Adicionar contenido.

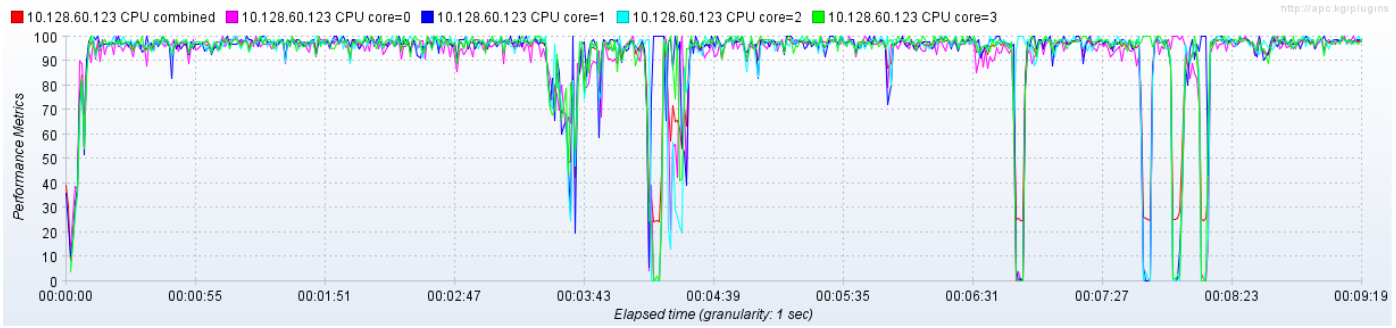

Figura 35: Utilización del CPU 40 usuarios caso de prueba Adicionar contenido.

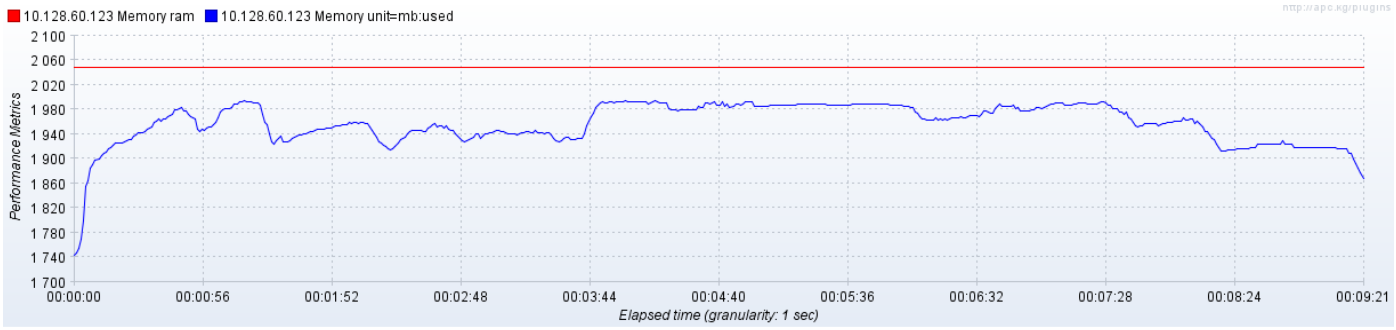

Figura 36: Utilización de la memoria 40 usuarios caso de prueba Adicionar contenido.

Caso de prueba Eliminar contenido

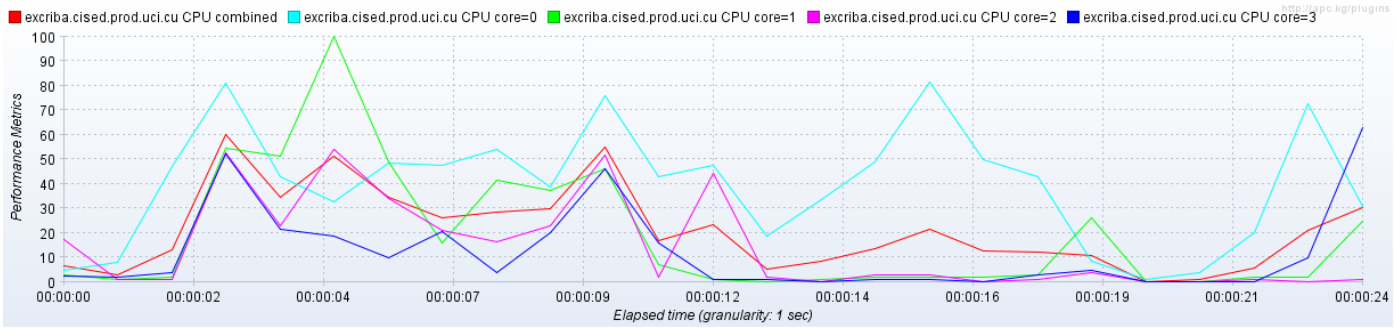

Figura 37: Utilización del CPU un usuario caso de prueba Eliminar contenido.

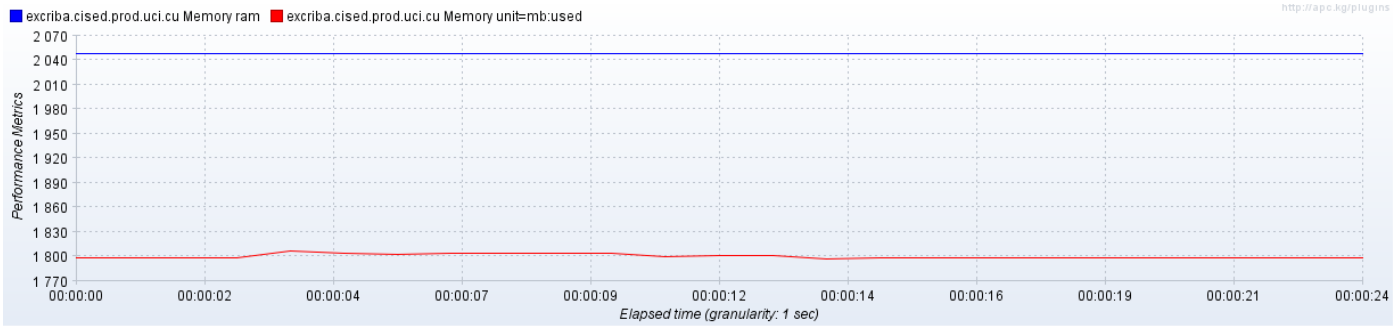

Figura 38: Utilización de la memoria un usuario caso de prueba Eliminar contenido.

Figura 39: Utilización del CPU 40 usuarios caso de prueba Eliminar contenido.

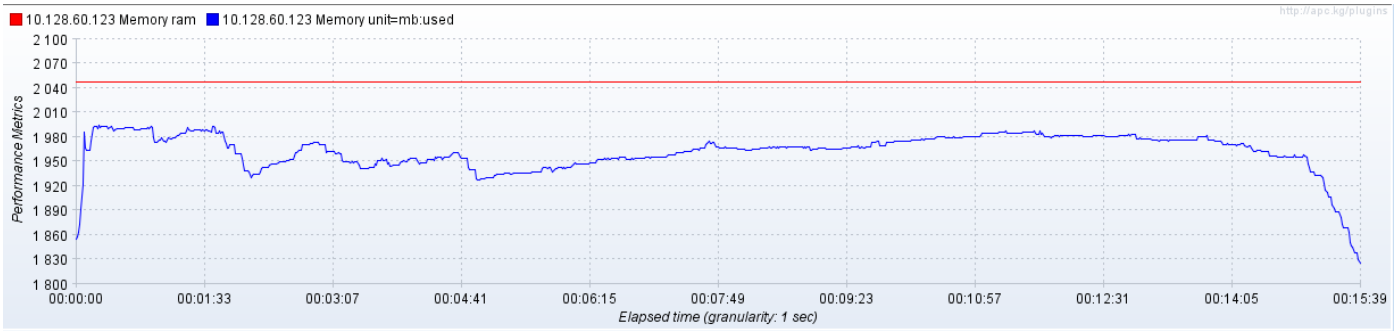

Figura 40: Utilización de la memoria 40 usuarios caso de prueba Eliminar contenido.

Anexo 7: Utilización de los recursos en el Centro CDAE

Caso de prueba Adicionar contenido

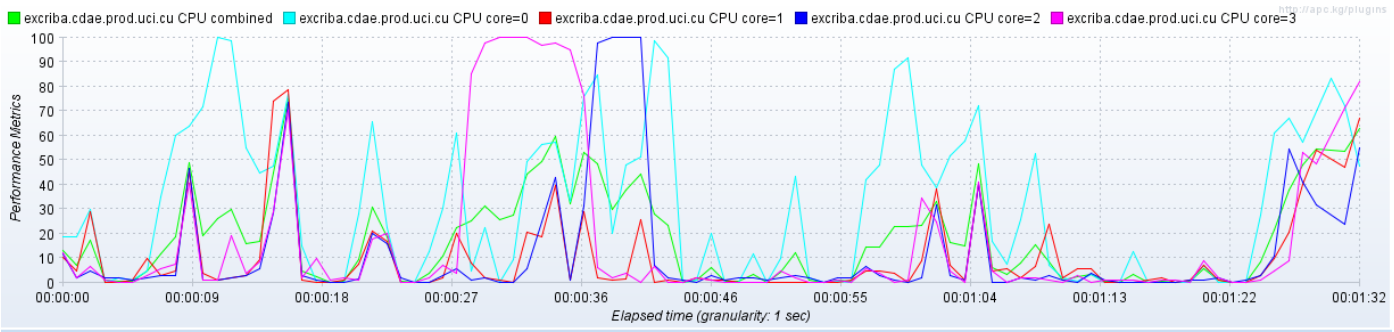

Figura 41: Utilización del CPU un usuario caso de prueba Adicionar contenido.

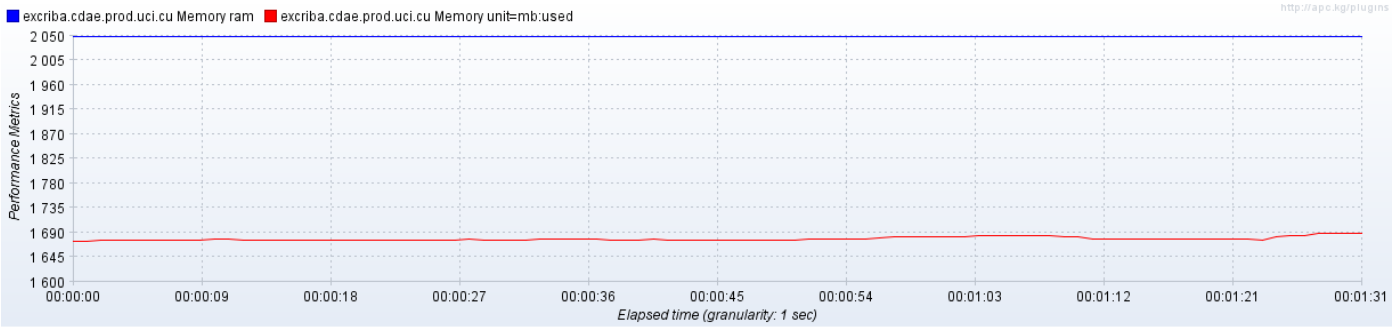

Figura 42: Utilización de la memoria un usuario de prueba Adicionar contenido.

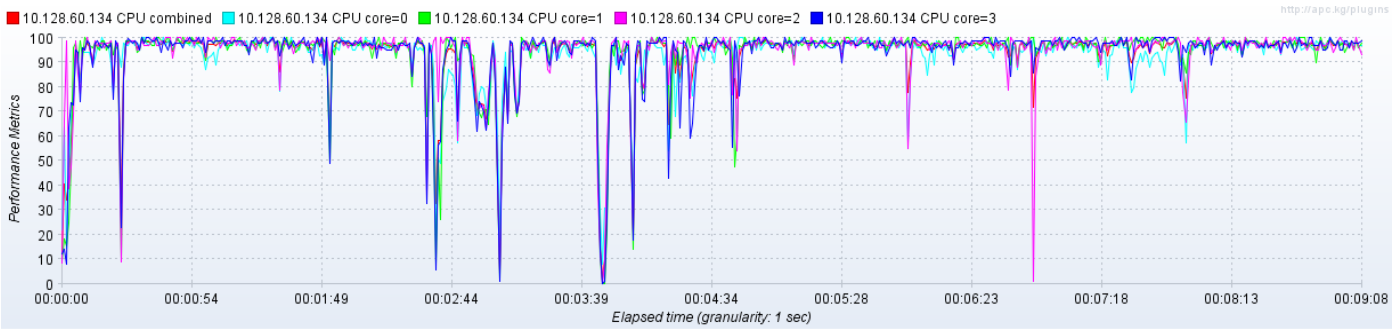

Figura 43: Utilización del CPU 40 usuarios caso de prueba Adicionar contenido.

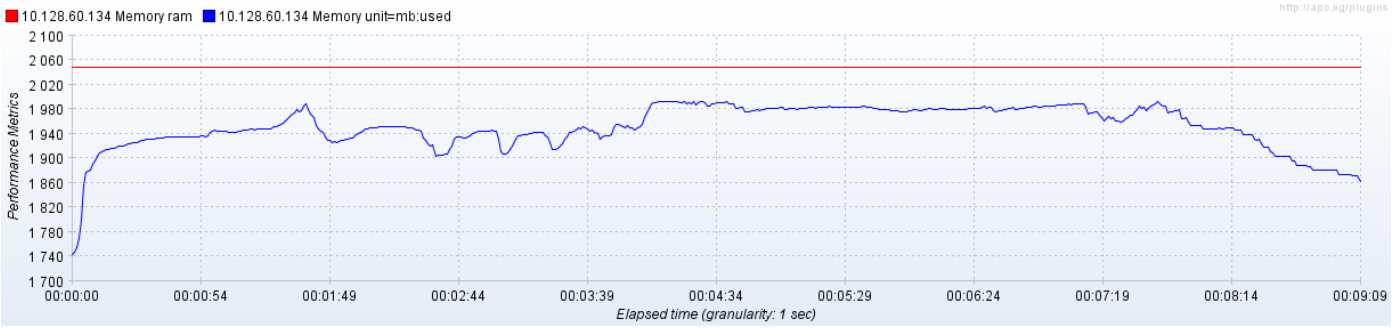

Figura 44: Utilización de la memoria 40 usuarios caso de prueba Adicionar contenido.

Caso de prueba Eliminar contenido

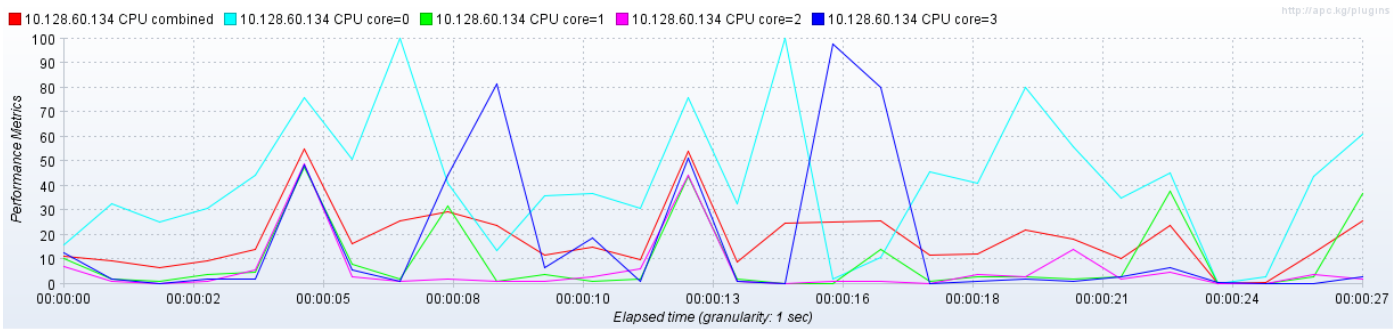

Figura 45: Utilización del CPU un usuario caso de prueba Eliminar contenido.

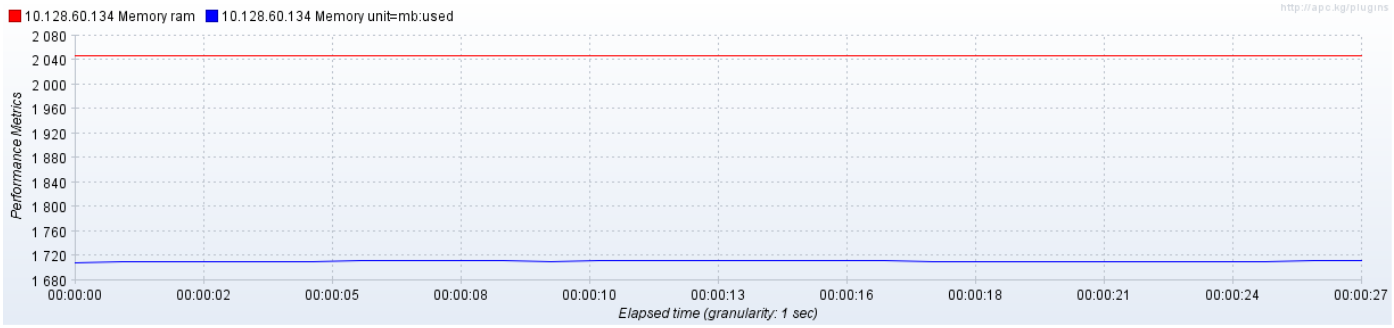

Figura 46: Utilización de la memoria un usuario caso de prueba Eliminar contenido.

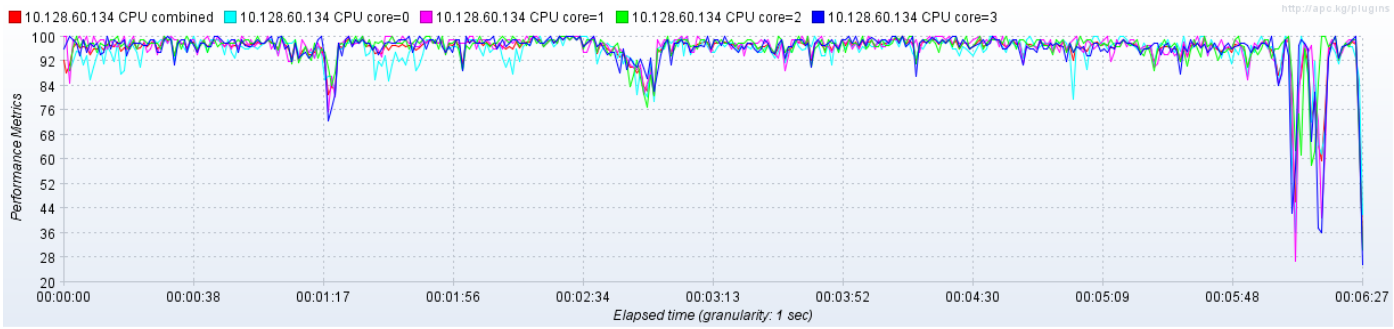

Figura 47: Utilización del CPU 40 usuarios caso de prueba Eliminar contenido.

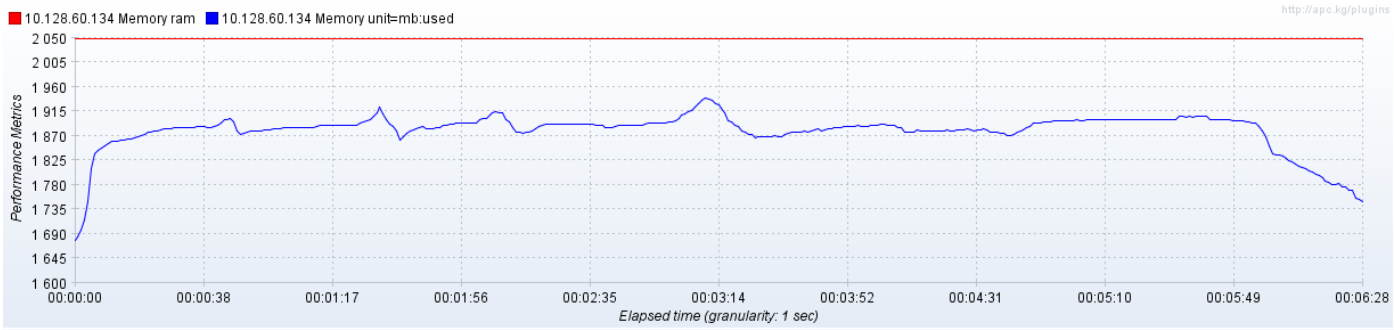

Figura 48: Utilización de la memoria 40 usuarios caso de prueba Eliminar contenido.

Anexo 8: Utilización de los recursos en el Centro CEGEL

Caso de prueba Adicionar contenido

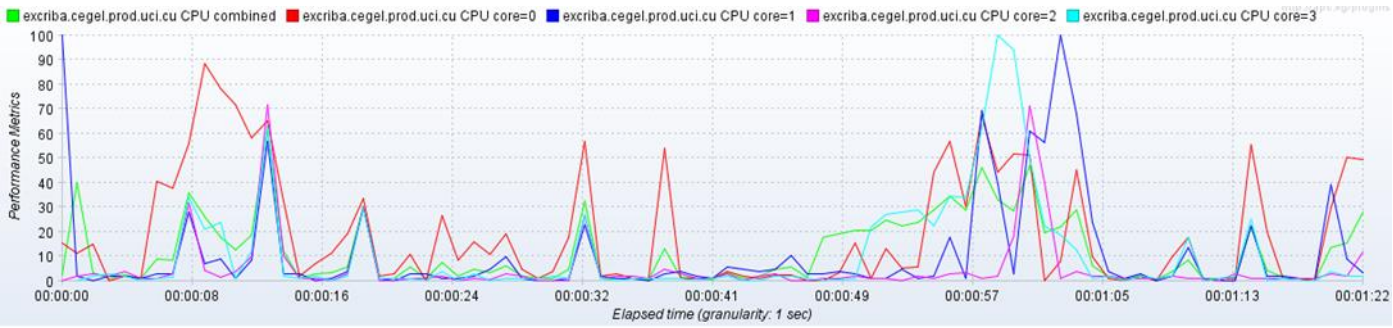

Figura 49: Utilización del CPU un usuario caso de prueba Adicionar contenido.

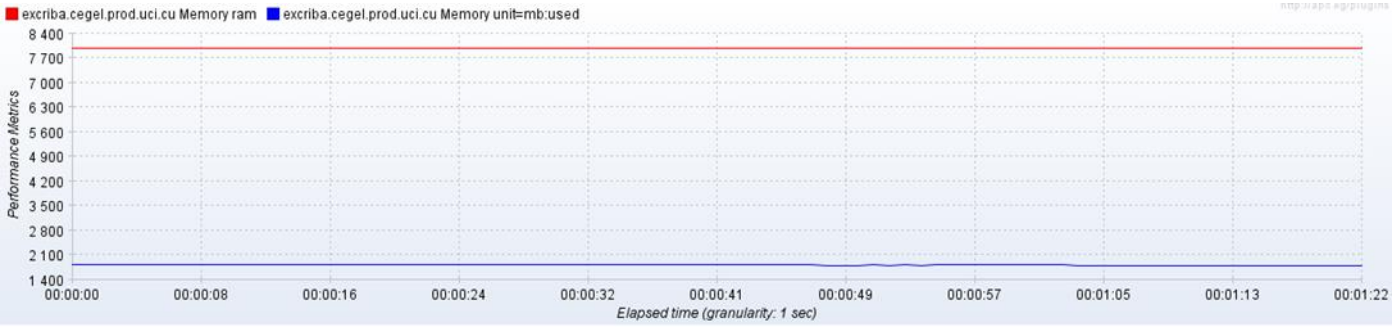

Figura 50: Utilización de la memoria un usuario caso de prueba Adicionar contenido.

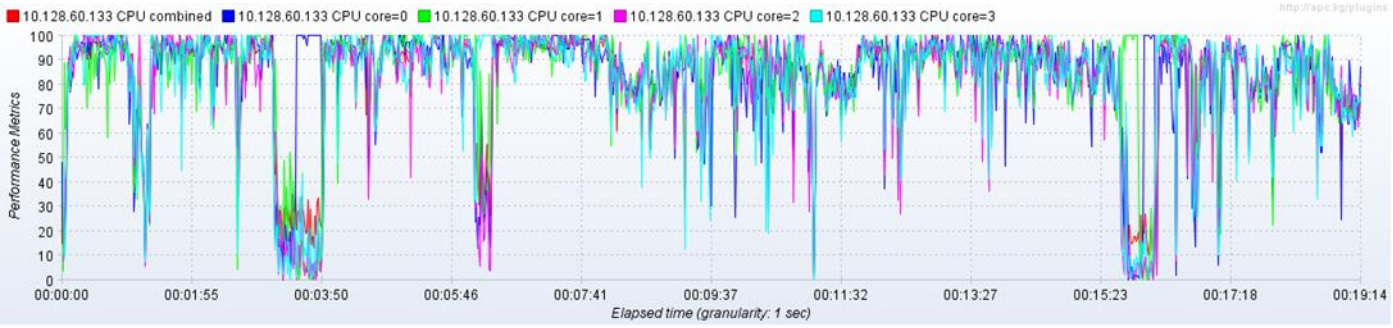

Figura 51: Utilización del CPU 40 usuarios caso de prueba Adicionar contenido.

Figura 52: Utilización de la memoria 40 usuarios caso de prueba Adicionar contenido.

Caso de prueba Eliminar Contenido

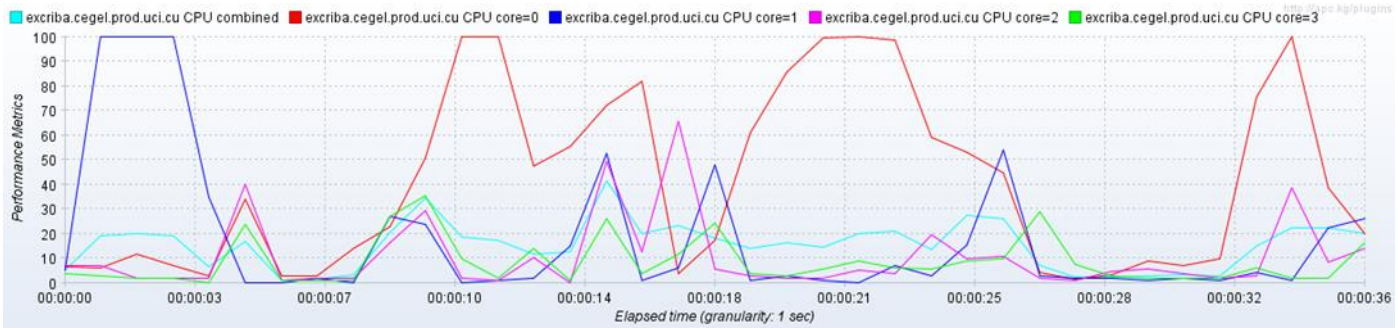

Figura 53: Utilización del CPU un usuario caso de prueba Eliminar contenido.

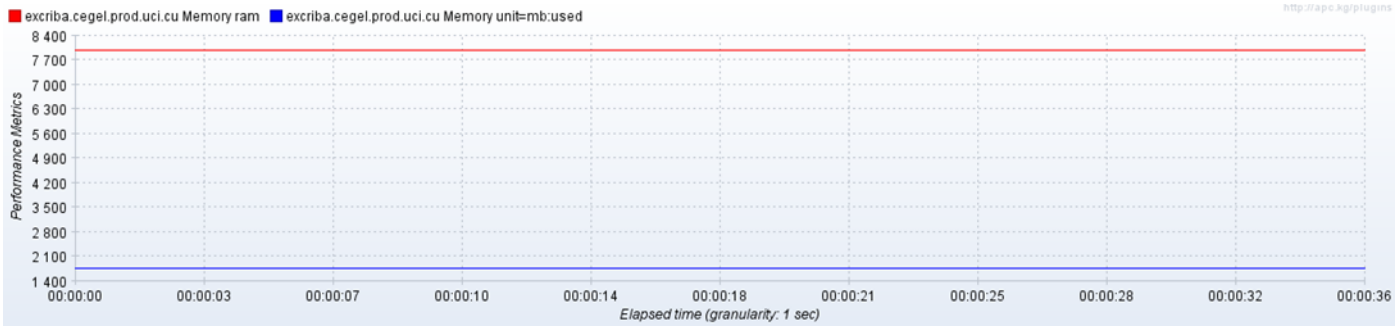

Figura 54: Utilización de la memoria un usuario caso de prueba Eliminar contenido.

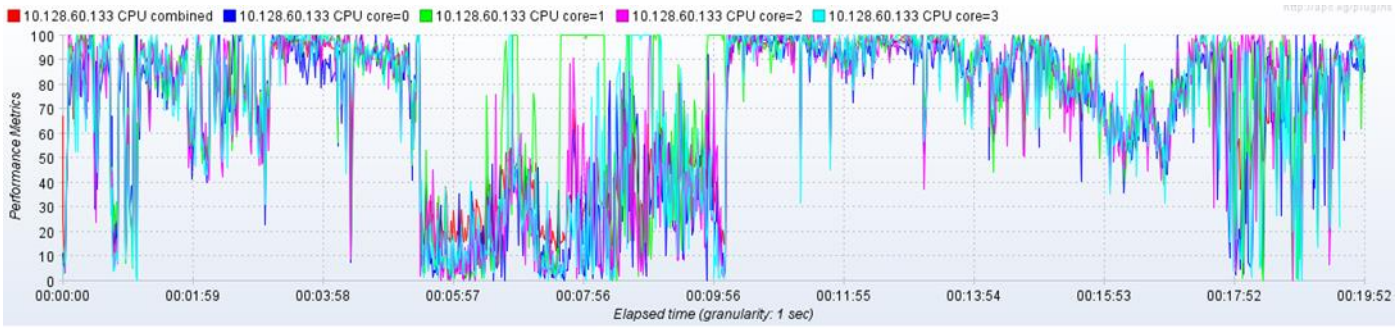

Figura 55: Utilización del CPU 40 usuarios caso de prueba Eliminar contenido.

Figura 56: Utilización de la memoria 40 usuarios caso de prueba Eliminar contenido.

Anexo 9: Utilización de los recursos en el Centro CISED después del ajuste

Caso de prueba Adicionar contenido

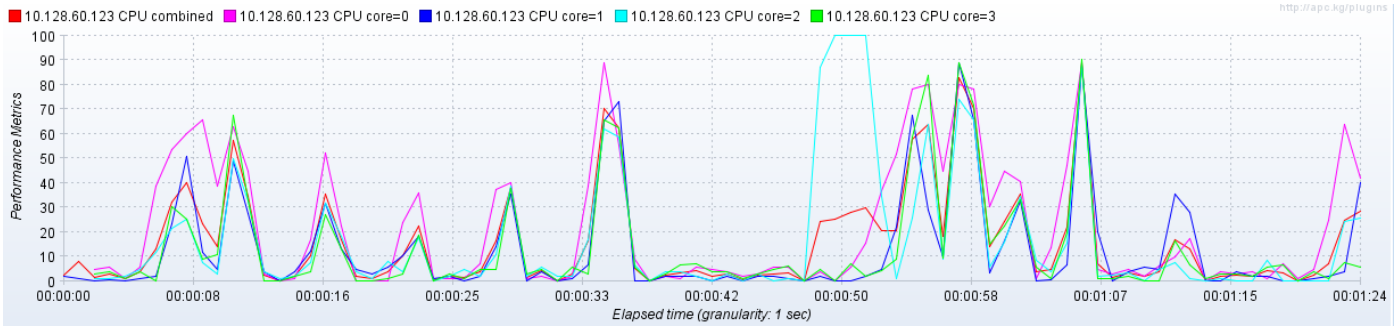

Figura 57: Utilización del CPU un usuario caso de prueba Adicionar contenido.

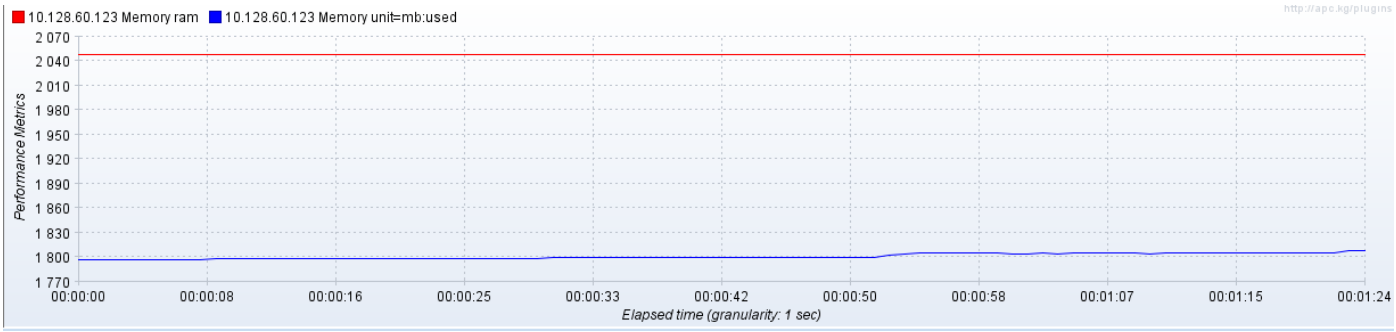

Figura 58: Utilización de la memoria un usuario caso de prueba Adicionar contenido.

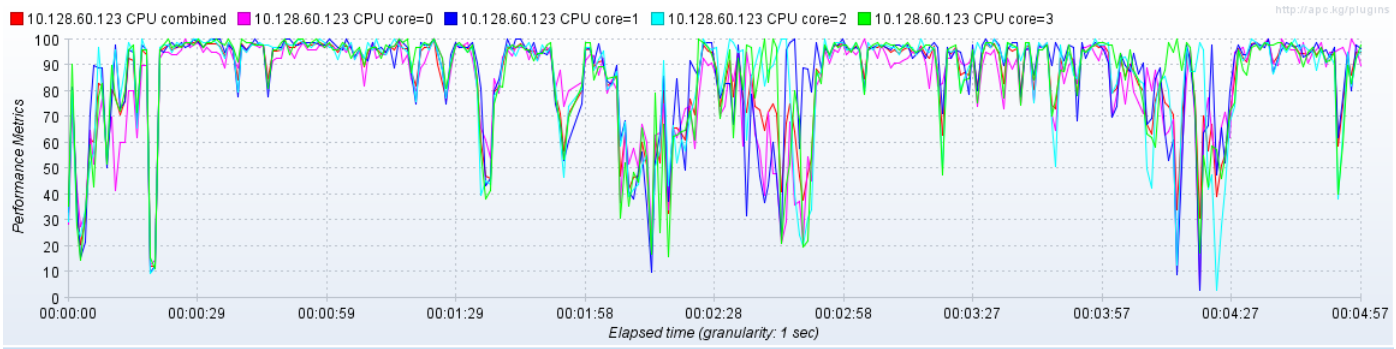

Figura 59: Utilización del CPU 40 usuarios caso de prueba Adicionar contenido.

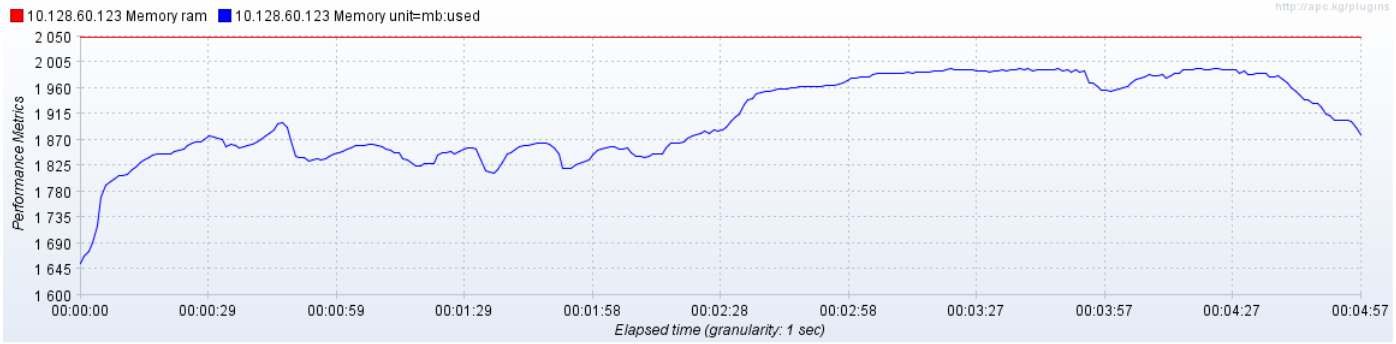

Figura 60: Utilización de la memoria 40 usuarios caso de prueba Adicionar contenido.

Caso de prueba Eliminar contenido

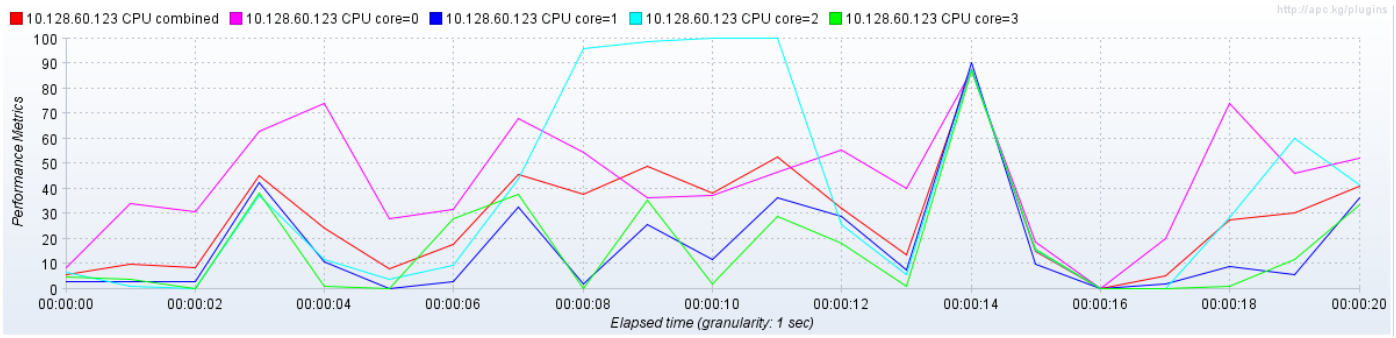

Figura 61: Utilización de la memoria un usuario caso de prueba Eliminar contenido.

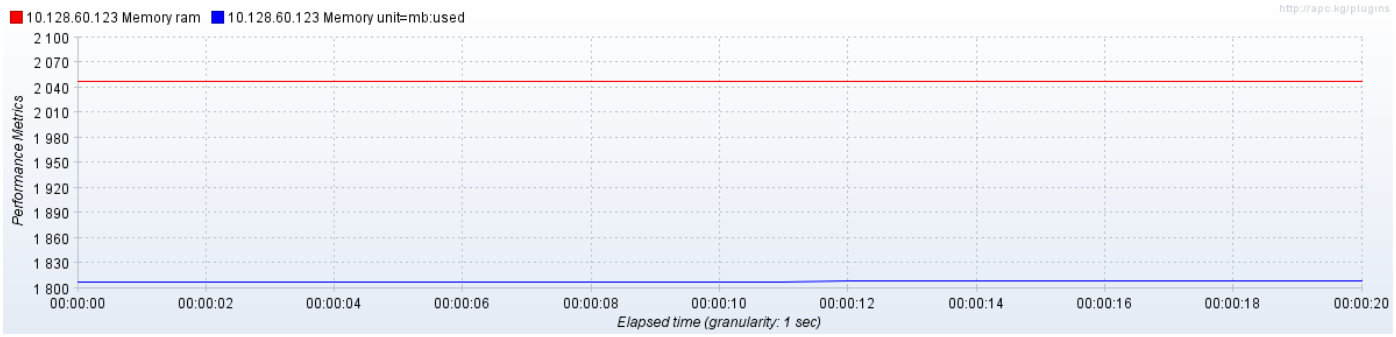

Figura 62: Utilización de la memoria un usuario caso de prueba Eliminar contenido.

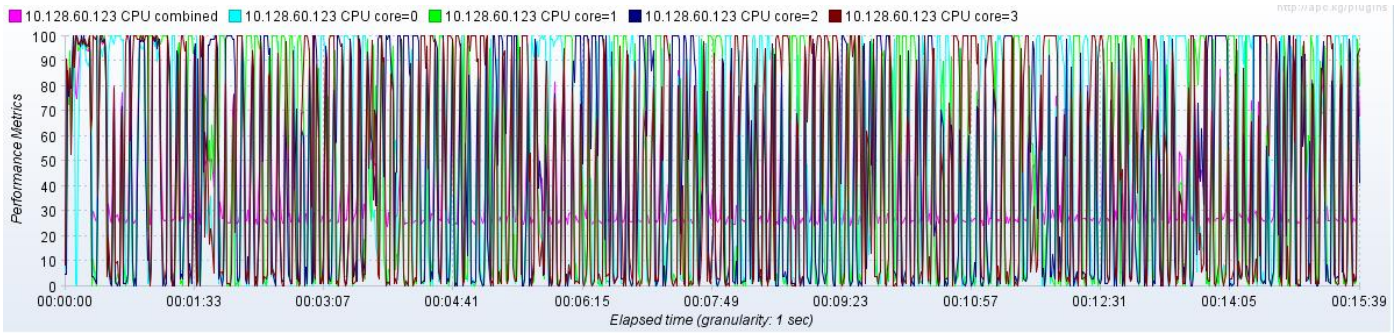

Figura 63: Utilización de la disco 40 usuarios caso de prueba Eliminar contenido.

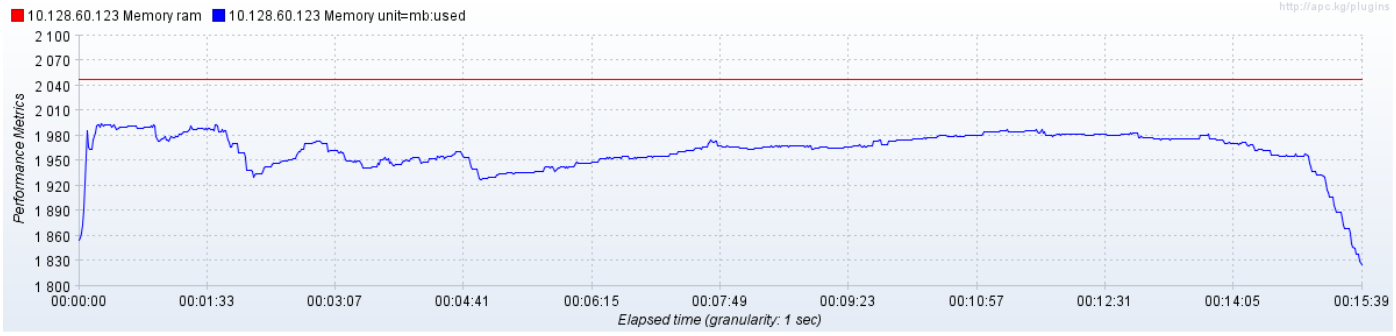

Figura 64: Utilización de la memoria 40 usuarios caso de prueba Eliminar contenido.

Anexo 10: Utilización de los recursos en el Centro CDAE después del ajuste

Caso de prueba Adicionar contenido

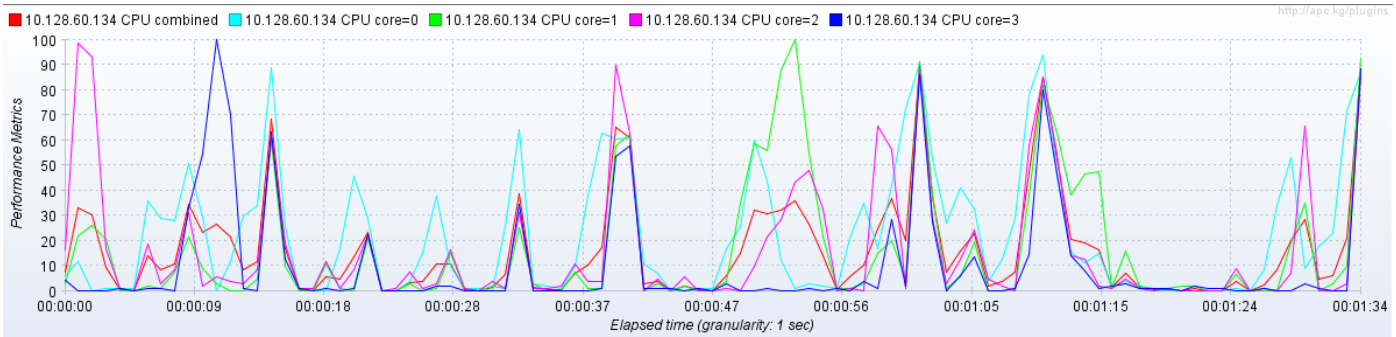

Figura 65: Utilización del CPU un usuario caso de prueba Adicionar contenido.

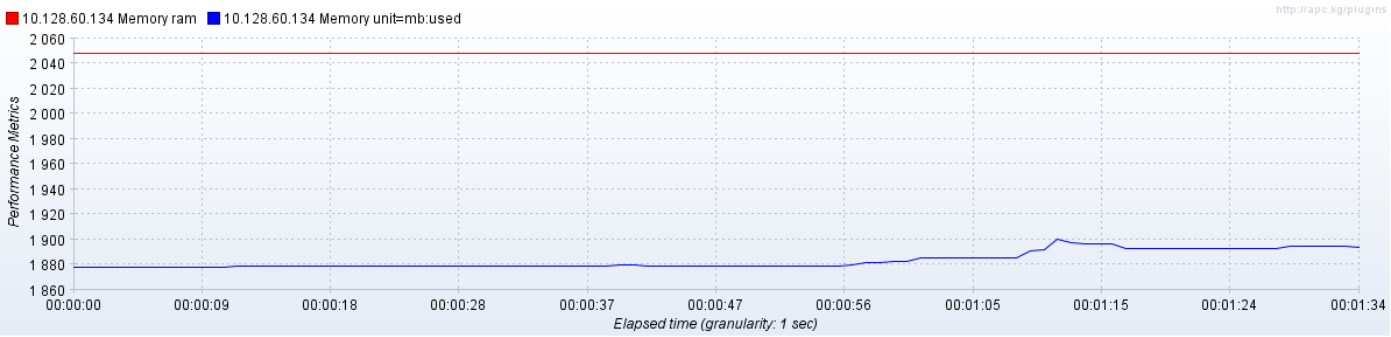

Figura 66: Utilización de la memoria un usuario caso de prueba Adicionar contenido.

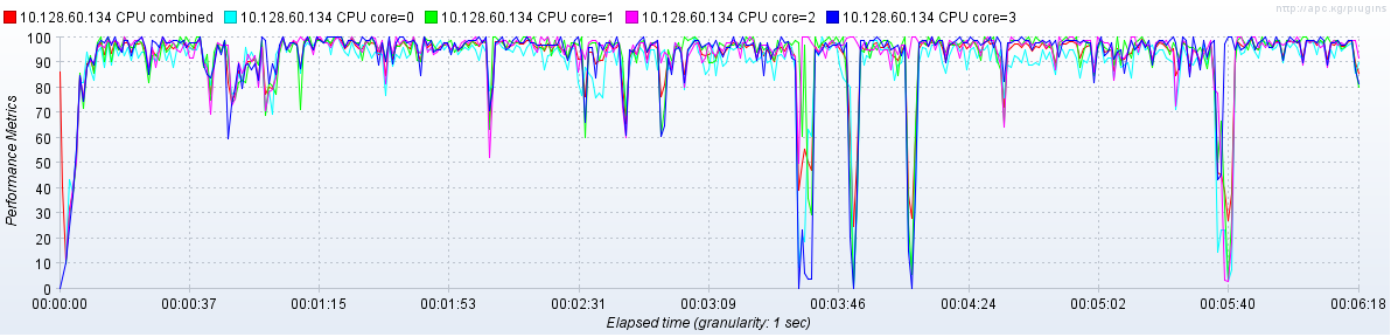

Figura 67: Utilización del CPU 40 usuarios de prueba Adicionar contenido.

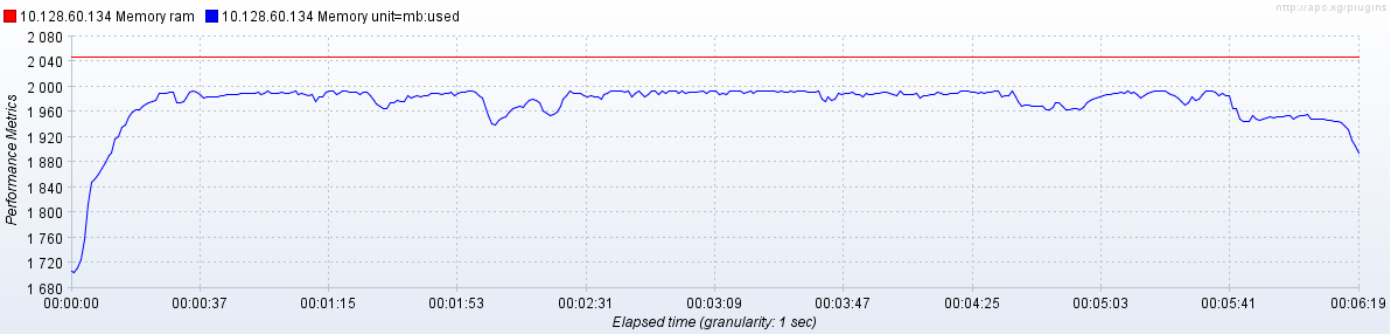

Figura 68: Utilización de la memoria 40 usuarios caso de prueba Adicionar contenido.

Caso de prueba Eliminar contenido

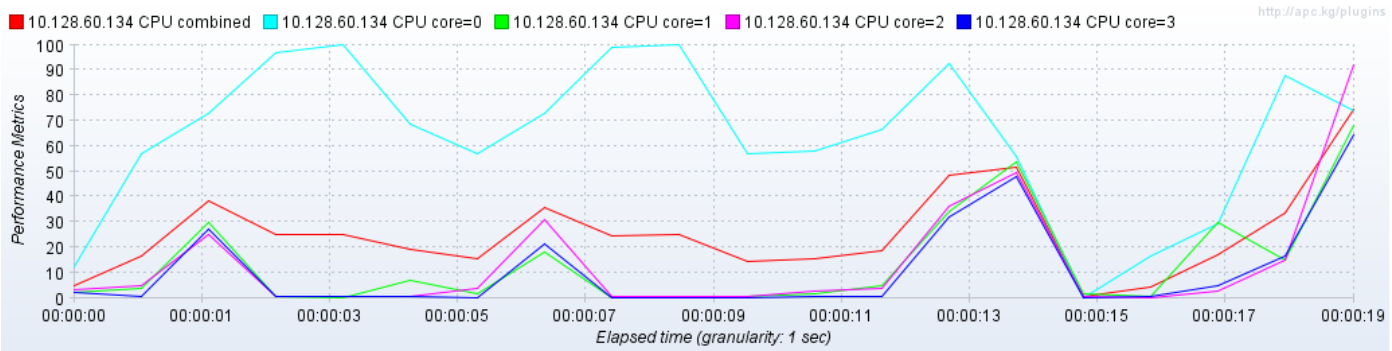

Figura 69: Utilización del CPU un usuario caso de prueba Eliminar contenido.

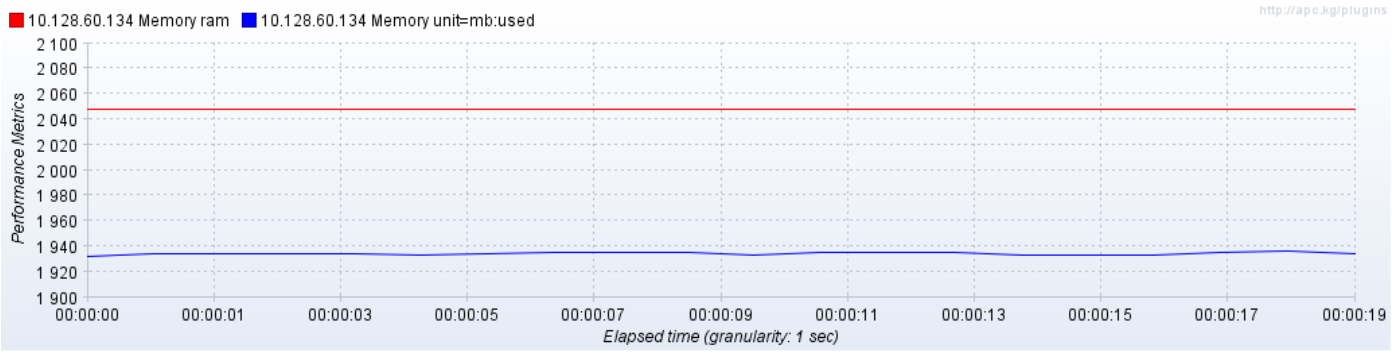

Figura 70: Utilización de la memoria un usuario caso de prueba Eliminar contenido

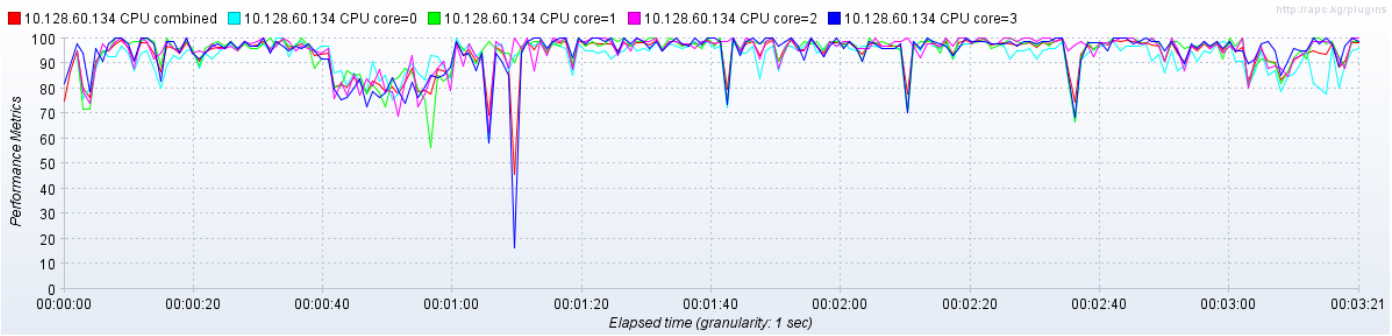

Figura 71: Utilización del CPU 40 usuarios caso de prueba Eliminar contenido.

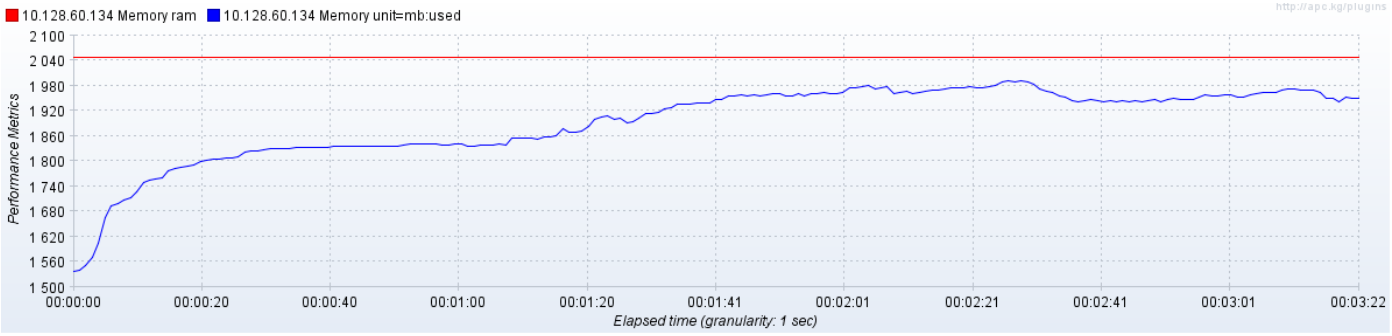

Figura 72: Utilización de la memoria 40 usuarios caso de prueba Eliminar contenido.