INSTITUTO TECNOLÓGICO DE COSTA RICA

ESCUELA DE INGENIERÍA EN COMPUTACIÓN

PROGRAMA DE MAESTRÍA

Desarrollo de un modelo para el análisis de pruebas de rendimiento de software en ambientes de Big Data, para identificar patrones mediante reglas de asociación

Tesis para optar al grado de

Magister Scientae en Computación

Jorge Mauricio Acuña Gómez.

Cartago, Costa Rica

Noviembre - 2015

Abstract

The application development process considers several stages or phases in the product life cycle, there are different variations of the kind of the stages the lilfe cycle should follow, but the testing phase is always present. In a testing stage, the decision if the application goes live or not is made.

In this phase, the application and its new capabilities or functionalities are tested, in order to guarantee it is working properly and as expected. There are different kind of tests, among them, the functional and performance testing. Functional testing is in charge of validating that all the processes developed are producing the correct output, it means that everything is working as expected. In the other hand, the performace testing ensures that the application performance is optimal, that means the system can process and attend all the user or application requests without any complication.

The performance testing has as main objective to demonstrate that the hardware and software work properly together, according to the loads like concurrent processes or users in production environments. Because of that, this kind of testing is so important. Applying performance testing, allows the development teams to identify potenctial interruptions or future failures in production systems.

The analysis and evaluation of the obtained results from this kind of testing can be tedious and complex, requiring the investment of time, money and human resources. For this reason, it is important to conduct studies on how to improve the involved processes in the analysis of these results.

The present study proposes and develops a model, that using data mining techniques such as the generation of association rules, presents a series of standards and steps that must be met, to facilitate development teams the analysis and evaluation of the results.

The model, also, presents another significant challenge, and it is that currently we must address the data processing more closely, as we face a huge amount of data with complex

structure and different kind of formats. It is for this reason that one of the objectives of this work is to take advantage of the available options for Big Data today.

In this way, using the model and attacking the identified challenges, the decisions can be made more efficiently and effective.

Resumen

Cuando se trata de desarrollar un sistema informático se deben considerar ciertas etapas o fases dentro del ciclo de vida de la aplicación, existen variantes sobre las etapas que debe contener, pero siempre existe una de ellas que es la responsable de proyectar los resultados para tomar la decisión de aprobar o no ciertas funcionalidades o requerimientos incluidos en el sistema.

Esta etapa es donde se llevan a cabo las pruebas de la aplicación, existen diferentes tipos de pruebas, como las funcionales y las de rendimiento o desempeño. Las pruebas funcionales buscan garantizar que los procesos funcionan o producen los *outputs* esperados, por otro lado, las pruebas de desempeño buscan garantizar que el rendimiento de los procesos sea óptimo.

Las pruebas de rendimiento tienen el objetivo primordial de demostrar que el hardware y el software funcionan adecuadamente de acuerdo a la carga a la que será expuesta, de ahí la importancia de este tipo de pruebas. Realizando estas pruebas se podrán evitar interrupciones o fallas futuras en los sistemas de producción.

El análisis y evaluación de los resultados arrojados por este tipo de pruebas puede llegar a ser tedioso y complicado, requiriendo la inversión de tiempo, dinero y recurso humano. Es por esta razón, que es importante realizar estudios sobre cómo mejorar los procesos involucrados en el análisis de los resultados de estas pruebas.

El presente trabajo, tiene como objetivo desarrollar un modelo, que mediante la utilización de técnicas de reglas de asociación, presente una serie de normas y pasos que deben ser cumplidos para facilitar a los equipos de desarrollo el análisis y evaluación de los resultados de las pruebas.

El desarrollo del modelo, presenta otro reto significativo, y es que actualmente se debe tratar el tema del procesamiento de los datos con más atención, ya que nos enfrentamos a grandes cantidades de datos, con una estructura y formatos variados y complejos. Es por

esto, que otro de los objetivos planteados en este trabajo es el tomar ventaja de las opciones disponibles hoy en día para *Big Data*.

De esta forma, por medio del modelo se busca tomar decisiones de una manera más eficiente y eficaz.

www.tec.ac.cr

TEC | Tecnológico de Costa Rica
Programa de Maestría en Computación
Teléfono: 2550-2402

ACTA DE APROBACION DE TESIS

Con fundamento en lo que establecen los **Artículos 22-24-25** del "Manual de Normas y Procedimientos para optar por el título de "MAGÍSTER SCIENTIAE EN COMPUTACION", el Tribunal Examinador de Tesis (TET), nombrado con el propósito de evaluar la tesis de grado.

"Desarrollo de un modelo para el análisis de pruebas de rendimiento de software en ambientes de Big Data, para identificar patrones mediante reglas de asociación."

Habiendo analizado el resultado general del trabajo presentado por el estudiante:

Primer Apellido	Segundo Apellido	Nombre	No. de carné
Acuña	Gomez	Jorge Mauricio	201174070

Emite el siguiente dictamen:

O APROBADO

El TET, considerando que el trabajo realizado por el estudiante es SOBRESALIENTE, le otorga la siguiente MENCION HONORIFICA:

O CUM LAUDE O MAGNA CUM LAUDE O SUMMA CUM LAUDE

O REPROBADO

O SE RECOMIENDA O NO SE RECOMIENDA

Brindarle una nueva oportunidad para la DEFENSA PUBLICA de su Tesis

NUEVA FECHA: _____

Dando fe de lo aquí expuesto firmamos (*IDEM*: HOJAS DE APROBACION DE TESIS)

Dr. Carlos González Alvarado
Profesor Asesor

Dr. José Enrique Araya Monge
Profesor Lector

MSc. Gustavo Montealegre Castro
Profesional Externo

Dr. Roberto Cortés Morales
Coordinador del Programa
de Maestría en Computación
TEC | Tecnológico de Costa Rica
Maestría en Computación

19 de noviembre de 2015

Sello

FT-07-MC

www.tec.ac.cr

TEC | Tecnológico de Costa Rica
Programa de Maestría
en Computación
Teléfono: 2550-2402

APROBACIÓN DE LA TESIS

"Desarrollo de un modelo para el análisis de pruebas de rendimiento de software en ambientes de Big Data, para identificar patrones mediante reglas de asociación"

TRIBUNAL EXAMINADOR

Dr. Carlos González Alvarado
Profesor Asesor

Dr. José Enrique Araya Monge
Profesor Lector

MSc. Gustavo Montealegre Castro Profesor Externo

Dr. Roberto Cortés Morales
Coordinador del Programa
de Maestría en Computación

Noviembre, 2015

Dedicatoria

A mis papás Jorge y Marielos.

Que con su trabajo y dedicación me ayudaron a ser profesional, y sin su esfuerzo no estaría hoy finalizando esta tesis.

A mi esposa Xinia.

Por ayudarme siempre y apoyarme durante este proceso de estudios, inclusive durante el nacimiento de nuestra hija Jimena. Sin su apoyo no lo hubiera logrado.

Finalmente a mi hija Jimena.

Para que siempre trabaje por sus sueños y deseos. Que si se desean de verdad, con esfuerzo, sacrificio y dedicación se pueden cumplir.

Agradecimientos

Primero que todo a Dios.

Quiero darle las gracias por permitirme estudiar, por darme la salud y acompañarme siempre durante el proceso de realización y finalización de este trabajo.

Al Dr. Carlos González Alvarado.

Por haber sido mi profesor y tutor, por haberme dado la guía y enseñado como realizar un trabajo de investigación.

A Gustavo Montealegre,

Por acceder a ser mi lector externo.

A Marisela Rodríguez,

Por su colaboración en la revisión de este documento.

Índice

Índice de Figuras

Índice de Tablas

CAPÍTULO 1 Introducción

El ciclo de vida de desarrollo de una aplicación, contempla como una de sus fases o etapas, las pruebas del sistema desarrollado. Estas pruebas pueden ser, pruebas funcionales o de desempeño. Contar con un buen plan de pruebas ayuda a los equipos de trabajo a detectar cualquier problema o anomalía en la aplicación antes de hacer el *release* del sistema a producción.

Las pruebas de rendimiento o desempeño tienen el objetivo de garantizar que la aplicación soporta en términos de rendimiento las cargas a las que se verá expuesta una vez que se haya puesto en marcha.

Dichas pruebas deben contemplar distintos aspectos de los elementos involucrados en cualquier sistema informático; de los cuales podemos nombrar la respuesta del sistema ante la ejecución de ciertas funcionalidades o procesos implementados y el estado de los componentes de hardware donde correrá la aplicación. Estos aspectos mencionados, anteriormente, son los que se utilizaron a la hora de medir o evaluar cada componente del sistema, ya sea de hardware o de software.

La aplicación de este tipo de pruebas es fundamental para los equipos de desarrollo actuales, ya que pueden evitar la pérdida en términos de tiempo, recurso humano y dinero a las empresas. La detección de una "pulga" o falla en un sistema puede llegar a representar gastos importantes de recursos, dependiendo del negocio y la aplicación involucrada.

Debido a eso, se determinó como una oportunidad el mejorar los procesos de análisis y evaluación de las pruebas de rendimiento de software.

La presente investigación tiene como objetivo principal proponer un modelo que permita mejorar la evaluación y el análisis de los resultados de estas pruebas.

La función de identificar problemas potenciales en los sistemas informáticos, luego de correr las pruebas puede ser muy compleja, dependiendo de la aplicación y su arquitectura;

esto conlleva a la inversión de recursos y tiempo para evaluar dichas pruebas, por lo que la aplicación del modelo sería de gran ayuda para el desarrollo y puesta en marcha de un sistema.

Por otro lado, al final de estas pruebas se cuenta con una gran cantidad de archivos o información proveniente de diferentes fuentes, las cuales pueden ser computadoras clientes, servidores o las diferentes aplicaciones con las que podría interactuar la aplicación, y que son importantes para ser consideradas en las pruebas de desempeño.

Uno de los retos de la investigación es que el modelo propuesto sea escalable, por lo que el enfoque de la misma es en ambientes de *Big Data*, pues se pueden llegar a manipular datos masivos y complejos.

Finalmente, lo que se persigue con el modelo es la identificación de patrones en los datos generados o procesados por medio de la obtención de reglas de asociación, pues por medio de ellas, se identificará un conjunto de relaciones presentado con cierta frecuencia en los datos producidos durante la aplicación de las pruebas. Este conjunto de reglas, indicará el comportamiento de la aplicación, el cual, dependiendo de los parámetros establecidos y resultados esperados, determinará si el sistema informático es adecuado o no para ser puesto en marcha.

El modelo propuesto hará que el proceso de análisis de pruebas sea más eficiente y eficaz, evitando, además, el error humano al realizar el análisis.

Por medio de este modelo, también se busca que los equipos de desarrollo puedan dedicar su tiempo a otras tareas y de esta forma ayudar al desarrollo de los proyectos en las organizaciones.

1.1 Definición del Problema

El análisis del resultado de las pruebas de rendimiento, es una de las etapas durante el ciclo de vida de desarrollo de los sistemas que se debe mejorar, pues en este proceso se invierte gran cantidad de recursos y tiempo para determinar si las pruebas fueron exitosas o no.

Por lo tanto, este análisis también busca la identificación de patrones o escenarios en los que el rendimiento de los sistemas se pueda ver comprometido.

El análisis de los datos recolectados de las pruebas de rendimiento es requerido para mejorar aspectos de confiabilidad, disponibilidad, desempeño y escalabilidad en los sistemas informáticos que se utilizan actualmente en las organizaciones. A estas pruebas de rendimiento se les conoce como pruebas de RAPS del inglés "*Reliability, Availability, Performance and Scalability*".

La propuesta para la identificación y análisis impactaría directamente en la disminución de los costos y el tiempo invertido actualmente para la evaluación de los resultados de las pruebas.

Como se ha mencionado, es importante tomar en cuenta que el proceso involucra el análisis de grandes cantidades de datos, por lo cual es necesario optimizar dicho proceso y tomar en cuenta opciones y herramientas que permitan el procesamiento de datos masivos y complejos. Para este tipo de procesamiento, es necesario contar con una plataforma que brinde la flexibilidad de ser modificada, permitiéndole adaptarse a las necesidades del procesamiento de los datos requeridas. La escalabilidad ofrece esa capacidad, por lo que la plataforma y herramienta de *Big Data* debe permitirlo, brindando la opción de ofrecer más capacidad y potencia de procesamiento, tanto a nivel horizontal como vertical. De esta forma, se garantiza que el modelo desarrollado contempla y soporta diferente carga de datos de acuerdo a la aplicación y plan de prueba propuesto.

Por último, la identificación de patrones en los resultados permitirá determinar si existe alguna condición o escenario que impacte o comprometa el desempeño de las aplicaciones; de esta manera, los equipos de desarrollo podrán tomar acciones correctivas en sus sistemas.

1.2 Justificación

Las pruebas de rendimiento son necesarias para cualquier aplicación que se desarrolle; además, son de suma importancia porque muestran donde nuestra aplicación o sistema puede ser mejorado. Por ejemplo, puede indicarnos bajo que escenarios de carga el sistema se encuentra comprometido al haber picos de utilización de procesador o memoria que se consideren perjudiciales para el funcionamiento del programa.

Los picos de memoria, podrían indicar que existe un *memory leak* en alguno de los procesos involucrados en los escenarios de prueba, lo que es muy dañino para la aplicación, ya que puede llegar a dejar fuera de servicio un servidor, lo que conlleva a la suspensión del servicio brindado. El problema de memoria, es considerado uno de los más importantes a considerar cuando se desarrollan sistemas.

La administración de los recursos de un servidor por parte del sistema informático debe ser adecuada; un buen manejo de ellos garantiza un buen desempeño.

Por otro lado, la correlación de las métricas recolectadas entre los datos del servidor como lo son: la memoria disponible, el disco duro, la red y el procesador utilizado vs los datos de la aplicación, como lo es el tiempo de respuesta para cada escenario de prueba aplicado; son importantes para comprender el comportamiento presentado que impacta en el desempeño de los sistemas informáticos.

La identificación de los escenarios de prueba o patrones se llevará a cabo por medio de algoritmos de asociación. La generación de las reglas de asociación ayuda a identificar relaciones entre los datos, los cuales indicarán los escenarios de prueba que comprometen la aplicación; o bien, si el sistema en cuestión funciona y soporta de manera adecuada las cargas y las pruebas a las que fue expuesto. Cada aplicación es distinta, por lo que es necesario del juicio experto para la definición de los escenarios de prueba, en este caso debe ser el equipo de desarrollo en conjunto con el usuario final los que los definan.

Además de la definición de los escenarios, se deben definir los resultados esperados, dependiendo de ello se determinará si el sistema informático está cumpliendo los parámetros esperados, y de esta manera soportando adecuadamente o no las cargas.

Logrando esto, los equipos de trabajo serán más ágiles; además, podrán dedicar recursos para otras tareas, aprovechando estos de mejor manera en un determinado momento del proyecto.

1.3 Objetivos

1.3.1 General

Desarrollar un modelo de análisis de pruebas de rendimiento de software en ambientes de *Big Data* para la identificación de patrones, ya sean positivos o negativos en los datos mediante reglas de asociación.

1.3.2 Específicos

1. Desarrollar un modelo que permita el análisis de las pruebas de rendimiento de sistemas informáticos.
2. Analizar y seleccionar la herramienta de *Big Data* más adecuada para el problema en cuestión.
3. Diseñar e implementar los procesos necesarios que permitan el procesamiento, reducción, recolección y preparación de los datos.
4. Utilizar reglas de asociación para la identificación de patrones en los datos recolectados de las pruebas.

1.4 Hipótesis

El desarrollo de un modelo para el análisis de pruebas de rendimiento de software, mediante herramientas de *Big Data* y reglas de asociación ayudan a la identificación de patrones que permitan la disminución de recursos invertidos en la fase de pruebas.

1.5 Antecedentes

1.5.1 Proceso actual para el análisis de las pruebas de rendimiento

Actualmente, las aplicaciones y su arquitectura son más complejas, lo que hace que la evaluación y análisis de las pruebas de rendimiento sean cada vez más complicadas, ya que se deben tomar en cuenta diferentes elementos, como las capas, donde nos referimos a aplicaciones o servicios que pueda contener un sistema.

Entre los elementos a evaluar se encuentran:

- la aplicación con sus escenarios de prueba y
- los diferentes componentes de hardware.

Los escenarios de prueban corresponden a cada funcionalidad o proceso definido en el sistema que se desea probar. Por ejemplo, podría ser un reporte, un cierre contable, un proceso de transformación de datos o el proceso de guardar los datos personales de un cliente. Cada uno de estos escenarios podría ejecutar uno o varios procesos, los cuales estarían involucrados en una transacción.

Por otro lado, los componentes de hardware de las computadoras o servidores donde corre la aplicación como el CPU, la memoria RAM, las solicitudes de I/O al disco duro y la transferencia de bytes por medio de la interface de red, indican cómo se está comportando el sistema ante las pruebas ejecutadas.

Para efectos de este proyecto se utilizaron datos de una aplicación crítica o de alta disponibilidad. Para esta se recolectó información tanto de los escenarios claves (previamente identificados) como del estado de los componentes del servidor.

Los datos del servidor se obtuvieron de información basada en los *performance counters* (Microsoft, Performance Counters, 2013). Además, se generaron logs con los tiempos de respuesta de los escenarios de prueba.

De acuerdo a los datos recolectados en los archivos de logs, se realizó un análisis con el fin de identificar potenciales “cuellos de botella” que pudieran comprometer la estabilidad de la aplicación en ambientes de producción. Reconocer estos es de suma importancia, ya

que ayuda a los equipos de desarrollo de software a solucionarlos antes de mover la aplicación a producción, evitando la interrupción de los sistemas o de los servicios, posteriormente.

Anteriormente, el proceso de dosificación, pre procesamiento, evaluación y análisis de los datos, por lo general se llevaba a cabo manualmente mediante **hojas electrónicas.** Realizar este proceso implica la inversión de gran cantidad de recursos y tiempo en la identificación de escenarios anómalos que puedan comprometer el desempeño de la aplicación.

Es importante mencionar que dicha herramienta tiene la limitación de que maneja un máximo de 1.048.576 filas. Uno de los objetivos de este tipo de pruebas es ejecutarlas por períodos largos de tiempo; por ello, la cantidad de información puede llegar a ser considerablemente grande y es una limitante de la herramienta para la elaboración de este tipo de análisis.

Como se mencionó anteriormente, el análisis y relación entre los datos generados por el servidor y la aplicación es realizado de forma manual. Lo que significa que si se identifica un pico durante un período de tiempo debería ser detectado manualmente por el equipo de trabajo, mediante el análisis de los datos almacenados en los archivos generados; esto se hace sumamente tedioso, además de que se necesita de mucho tiempo para obtener los resultados deseados.

Es importante mencionar que al ser un proceso relativamente manual es posible que se incurra en errores humanos al realizar el análisis de los datos, lo que provocaría tener resultados erróneos y no apegados a la realidad.

Por medio del modelo desarrollado se establecerían una serie de pasos y normas a seguir para facilitar el análisis de los resultados por medio de la utilización de herramientas de *Big Data,* que ayuden al procesamiento eficiente los datos, y la utilización de reglas de asociación para la detección de patrones en los datos producidos por las pruebas. Al definir pasos a seguir, implementar procesos para el pre-procesamiento de los datos y generar las reglas de asociación, se evitará la intervención y análisis manual de ciertos procesos, mejorando la evaluación de los resultados y evitando en gran medida el error humano.

1.5.2 Retos a enfrentar en la investigación

Inicialmente, se planteó el objetivo de estandarizar tanto el proceso de análisis de las pruebas de rendimiento como el de recolección de los datos, ya que esto permitirá la aplicación de los mismos procesos a las diferentes aplicaciones o sistemas.

Debido a la naturaleza de las pruebas, los datos generados podrían llegar a ser muchos y en esos casos se hablaría de Teras o Petabytes de información, dependiendo de la arquitectura y alcance de la aplicación. (6 Top Data Analysis Tools for Big Data)

Al hablar de *Big Data* se refiere a colecciones de datos masivos y complejos, donde el procesamiento de los datos se vuelve difícil al usar las herramientas tradicionales de administración de base de datos.

Resulta más complejo aún, porque el conjunto de datos es heterogéneo, debido a esta característica es complicado relacionar datos; además, la fuente de donde provienen los datos es diferente. Por otro lado, el procesamiento y dosificación de los datos debe ser ágil y eficaz (Stephen Kaisler, 2013).

El ciclo de vida a utilizar para el procesamiento de las datos será el CRISP-DM (Wikipedia, Cross Industry Standard Process for Data Mining). Este modelo ayuda a la comprensión, análisis y finalmente a implementar la solución propuesta.

Hay gran cantidad de herramientas que podemos utilizar para probar nuestras aplicaciones, como para probar cargas de usuarios conectados, *frameworks* para desarrollar y ejecutar pruebas de unidad, software que nos ayudan a verificar la cobertura de código que ha sido probado contra el que no ha sido probado, entre otros (Wikipedia, Category: Software testing tools). Todas ellas, tienen el objetivo de probar la aplicación generando resultados de manera aislada.

El enfoque de esta investigación, es mejorar el proceso de análisis de las pruebas de rendimiento, por medio de la correlación de los datos de las métricas de hardware (*performance counters*) con los escenarios funcionales definidos. La labor de relacionar estos datos es complicada, pues se estarían correlacionando pruebas funcionales con pruebas de rendimiento provenientes de distintas fuentes.

El modelo a desarrollar marcaría la pauta a seguir para mejorar el proceso, por medio de la utilización de reglas de asociación y herramientas de *Big Data*.

Para correlacionar los datos obtenidos de las diferentes fuentes (escenarios de prueba y métricas de hardware) se debe contemplar la implementación de procesos para la dosificación y preparación de los datos, y la aplicación de un algoritmo para la generación de las reglas de asociación.

La utilización del modelo, busca la detección eficaz y transparente de los escenarios o funcionalidades que degradan el *desempeño de la aplicación.*

Finalmente, lo que se busca con la identificación de este tipo de escenarios mediante técnicas de minería de datos y reglas de asociación, es reducir la inversión de recursos y tiempo. Como se ha mencionado anteriormente, las reglas de asociación ayudarán a la identificación de patrones en los datos. Los patrones indican la ocurrencia de escenarios donde ciertas métricas y pruebas son detectadas. De acuerdo a los patrones detectados y los resultados esperados se tomarán las decisiones correspondientes que definirán los siguientes pasos a seguir, ya sea resolver las fallas en la aplicación o aprobar la puesta en marcha del sistema.

CAPÍTULO 2 Marco Teórico

Para el desarrollo de la propuesta de optimización de las pruebas de rendimiento de software fue necesario recopilar información sobre qué son las pruebas de rendimiento, en qué consisten y cómo llevarlas a cabo. Es importante mencionar que uno de los aspectos importantes y novedosos de la propuesta es que debe ser escalable, ya que contempla la aplicación en ambientes de *Big Data*, por lo que también requirió del análisis y estudio de una de las herramientas más utilizadas hoy en día para procesamiento masivo de datos.

Por otro lado, se recopiló información sobre algunas propuestas existentes relacionadas a la investigación en cuestión, en las cuales se aplican algoritmos para la generación de patrones por medio de reglas de asociación, análisis de algoritmos, ente otros.

2.1 Pruebas de rendimiento

2.1.1 Definición de las pruebas de rendimiento

Para la creación de las *pruebas de desempeño* o *rendimiento* es necesario definir los siguientes aspectos:

a. **Escenario de prueba:** Serie lineal de eventos que nos ayuden a medir el desempeño de la aplicación. Un escenario describe una secuencia de acciones de usuario o comportamiento de la aplicación.
b. **Función:** Alguna funcionalidad que el software provee. Una función describe uno o más comportamientos o interacciones entre el usuario y la aplicación.
c. **Alcance:** Cada uno de los componentes o funcionalidades de la aplicación definen el alcance de las pruebas; por ejemplo, dependiendo de los nuevos requerimientos desarrollados o procesos automatizados, así se identificarán los escenarios o tipos de prueba necesarios a ejecutar. Asimismo, los diferentes elementos que componen la arquitectura del sistema también determinan la complejidad de los escenarios a

probar. Los componentes de la arquitectura podrían involucrar diferentes sistemas, interfaces o servicios.

d. **Medida:** La medida de la prueba puede ser cuantitativa o cualitativa. Un ejemplo de la medida cuantitativa es el tiempo de respuesta. La medida cualitativa es una descripción textual del desempeño de la prueba. Por ejemplo, una vez que tenemos el tiempo de respuesta podemos basarnos en éste para definirlo de manera cualitativa, como: Lento, rápido, etc.

2.1.2 Objetivos de las pruebas de rendimiento

De acuerdo a la página de software *testing* de Wikipedia (Wikipedia, Software performance testing), estos son los objetivos principales de las pruebas de rendimiento:

a. Demostrar que el sistema cumple con los criterios de desempeño.

b. Se pueden comparar diferentes sistemas para determinar cual tiene un mejor comportamiento o rendimiento.

Se pueden medir qué partes del sistema son las que se comportan mejor o cuales necesitan ajustes o mejoras.

2.2 Métricas

2.2.1 Métricas por utilizar para las pruebas de rendimiento

Basado en la definición y objetivos de las pruebas de rendimiento, y para efectos de esta investigación, se considera que las métricas por utilizar serán las siguientes:

- ***Información de la aplicación probada:***
    - Momento en que se ejecutó la prueba (*timestamp*).
    - Tipo de transacción o escenario de prueba.
    - Tiempo de respuesta.

- ***Performance Counters***
    - **Porcentaje de CPU utilizado:**
        - Indica qué porcentaje del procesador está siendo utilizado en determinado momento. (Tools, Windows Performance Counters Explained, 2015)
    - **Memoria disponible (en bytes):**
        - Indica cuántos bytes de la memoria están disponibles en un determinado momento. (Tools, Windows Performance Counters Explained, 2015)
    - **Interface de red (total de bytes/segundo):**
        - Indica la velocidad en bytes de como la interface de red está procesando los datos en un determinado momento. (Microsoft, Performance data for Bytes Total/sec counter, 2015)
    - **Disco duro (medido en bytes):**
        - Muestra el tamaño promedio en bytes de las solicitudes individuales de disco (tamaño de I/O) para un momento determinado. Por ejemplo, si el sistema tuvo 99 solicitudes de I/O de 8Kb y una solicitud de I/O de 2048 Kb, el promedio será de 28.4Kb. Cálculo = (8k*99) + (1*2048k) / 100. (Microsoft, Windows Performance Monitor Disk Counters Explained, 2015).

2.3 Herramienta para el procesamiento de datos masivos y complejos

Uno de los grandes retos que se presentan cuando se trata de procesar datos es la complejidad y gran cantidad de información disponible. Actualmente, se ha llegado al punto en donde gran cantidad de procesos han sido automatizados por medio de aplicaciones; sin embargo, dichos sistemas han generado demasiada cantidad de información y continúan haciéndolo, por lo que se vuelve indispensable contar con sistemas, algoritmos, técnicas o métodos eficientes que ayuden a procesar esa gran cantidad de datos.

Uno de los enfoques de este trabajo se realizó en herramientas de *Big Data open source*. En cuanto a costo, fines académicos y de investigación son más accesibles este tipo de herramientas.

Existen diferentes soluciones *open source* para procesar *Big Data*, aunque se ha enfocado el estudio sobre Hadoop. La elección para utilizar las herramientas de *Big Data* es indispensable en este estudio, ya que se busca una solución que sea escalable en el tiempo y que pueda ser utilizada como modelo para el análisis de las pruebas de rendimiento en diferentes sistemas informáticos.

2.3.1 Hadoop

Apache Hadoop es un *framework* de software que soporta aplicaciones distribuidas bajo una licencia libre. Permite a las aplicaciones trabajar con miles de nodos y Petabytes de datos. Hadoop se inspiró en los documentos Google para MapReduce y Google File System (GFS). (Hadoop)

Hadoop es una solución y opción que proporciona la escalabilidad deseada cuando se trata de procesar datos masivos y complejos, pues provee de procesamiento distribuido, además de trabajar con el paradigma *"Map/Reduce"*. Las características anteriores hacen de Hadoop un candidato idóneo para hacerlo parte del modelo propuesto en esta investigación.

2.3.1.1 **Distribución de Hadoop** (Handler B. D., 2015)

Existen varias distribuciones de Hadoop disponibles en el mercado, para efectos de este estudio se utilizo la distribución de *Cloudera*. Esta es una compañía que provee soluciones de software basadas en Apache/Hadoop; la distribución *open source* propuesta por la compañía a la que llamaron CDH del inglés *"Cloudera Distribution Hadoop"*, se enfoca en el desarrollo de esta tecnología para las empresas que requieran del procesamiento de *Big Data*. (Wikipedia, Cloudera, 2016)

Hadoop es un proyecto de alto nivel que está siendo construido y usado por una comunidad global de contribuidores, mediante el lenguaje de programación Java.

Hadoop es una *suite* de aplicaciones o herramientas. Como se puede mostrar en la figura 1, Hadoop está compuesto por diferentes herramientas, como *Pig, Hive*.

Figura 1. Distribución de Hadoop (Fuente: Big Data Handler web site)

A continuación se presentan los pros y los contras de Hadoop como herramienta de Big Data. La identificación de las ventajas y limitaciones de la herramienta ayuda a la comprensión y elección de la misma para trabajar en ambientes de Big Data.

2.3.1.2 **Limitaciones de Hadoop** (MetaScale, 2013)

Ente las limitaciones de Hadoop se pueden mencionar las siguientes:

1. No es una base de datos SQL de alta velocidad.
2. No es una tecnología particularmente simple.
3. No es fácil de conectar a los sistemas legacy.
4. No es un remplazo de data warehouses tradicionales. Es un producto complementario.
5. La arquitectura en torno a los datos (la manera de almacenar los datos, como se des-normalizan los datos, como se ingieren los datos, y la forma de extraer los datos) es diferente en Hadoop.
6. Conocimiento de Linux y Java es requerido.

2.3.1.3 **Principales Características de Hadoop** (Handler B. D., 2015)

Es importante tomar en cuenta y analizar las ventajas que presenta Hadoop como herramienta para procesar grandes cantidades de información; entre ellas se pueden nombrar las siguientes:

- **Escalabilidad:**
  - Nuevos nodos pueden ser agregados según la necesidad del proyecto sin modificar formatos de datos, el proceso de carga de los datos o cómo los *Jobs* fueron escritos.
- **Bajo Costo:**
  - Hadoop provee procesamiento masivo de datos distribuidos. El resultado es una disminución considerable en el costo por Terabyte de almacenamiento.
- **Flexibilidad:**
  - Brinda procesamiento de datos estructurados y no estructurados provenientes de diferentes o la misma fuente de datos, permitiendo un análisis más profundo y detallado de la información.
- **Confiabilidad:**
  - Al perderse uno de los nodos el sistema re-direcciona el trabajo a uno de los nodos disponibles, continuando con el procesamiento de los datos sin interrumpir el programa.

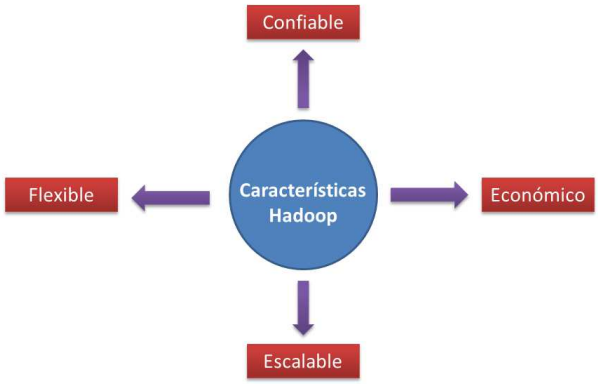

Figura 2. Principales Características de Hadoop.
(Fuente: Traducida de Big Data Handler web site)

Si bien es cierto hay limitaciones que se deben considerar antes de usar la herramienta, como las mencionadas anteriormente, se consideran más relevantes y de más provecho las ventajas que brinda Hadoop. La escalabilidad, el bajo costo, la confiabilidad y la flexibilidad son características importantes a la hora de la elección de la herramienta.

En esta investigación la solución debe ser escalable tanto entre las propuestas de software a analizar, como en el tiempo. Debe ser una solución válida conforme pasa el tiempo, donde el modelo propuesto en esta investigación sea una opción factible a considerar.

La posibilidad de poder incluir tantos nodos como se necesiten, hace de la solución una opción práctica y de fácil manejo.

Además, la capacidad de procesamiento de datos estructurados y no estructurados hace de esta opción una candidata a la hora de elegir la herramienta, ya que puede ser utilizada en diferentes ambientes y aplicaciones. Provee opciones para procesar los datos e integrar la información, estructurándola finalmente en los formatos deseados y requeridos para el análisis en cuestión.

El costo es un punto a favor, pues no hay que invertir dinero para utilizar la herramienta debido a que es *open source.* Convirtiéndola en una opción atractiva a la hora de decidir.

Finalmente, la confiabilidad brindada a la hora de procesar los datos es de suma importancia, ya que garantiza la redistribución del trabajo entre los nodos disponibles; haciendo el sistema propuesto tolerante, por ejemplo, ante las fallas de un determinado nodo.

2.4 Técnicas de minería de datos

2.4.1 Reglas de asociación

Como se menciona en la teoría, las reglas de asociación es una de las técnicas de minería de datos, se utilizan para descubrir hechos que ocurren en común dentro de un determinado conjunto de datos. (Sistemas de soporte a la toma de decisiones, 2010)

Las reglas de asociación son una poderosa técnica de minería de datos, son utilizadas para buscar reglas que revelen la naturaleza de las relaciones o asociaciones entre datos de las entidades. Las asociaciones resultantes pueden ser utilizadas para filtrar la información, para analizarlas y, posiblemente, para definir un modelo de predicción basado en la observación del comportamiento. (Juan Camilo Giraldo Mejía, 2012)

Para información más detallada sobre las reglas de asociación consultar (Wikipedia, Reglas de Asociacion).

Como se mencionó anteriormente, los datos de las pruebas de rendimiento serán obtenidos desde dos tipos de fuentes: la generada por el servidor (*performance counters*) y los datos generados por la aplicación encargada de estresar o probar el sistema.

El objetivo es identificar patrones o relaciones entre los datos generados por las dos fuentes. De esta manera, se podrán obtener conclusiones como:

- Al ejecutar el escenario A con X carga durante K tiempo el porcentaje de procesador oscila entre 70%-85%, lo que de acuerdo a los parámetros establecidos por el equipo de trabajo es un escenario negativo, pues está afectando el rendimiento de la aplicación. De esta forma, el equipo de desarrollo podrá enfocarse en mejorar el desempeño de las funcionalidades involucradas en el escenario A.

La identificación de este tipo de patrones se llevará acabo, finalmente, por la generación de las reglas de asociación aplicadas a los datos procesados y dosificados de las pruebas de rendimiento ejecutadas.

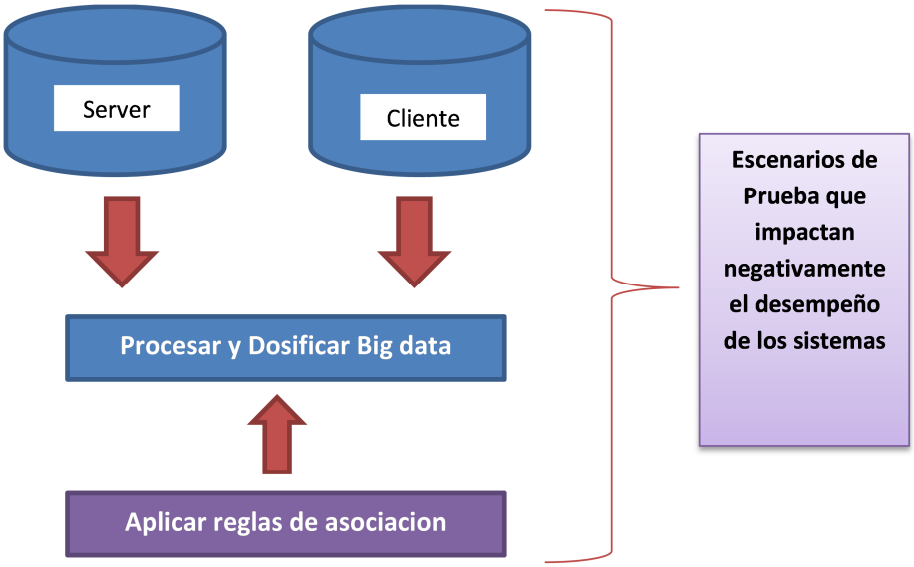

Figura 3. Descripción del proceso inicial propuesto para detección de escenarios negativos.

(Fuente: Creación propia).

2.4.2 Algoritmos utilizados

Información sobre los algoritmos tomada de (Introduction to Datamining); (Raudel Hernandez Leaon, 2010); (Apriori algorithm for Data Mining - made simple); (Analisis de asociaciones)

Apriori:

- Este algoritmo se usa en minería de datos para encontrar Reglas de asociación en un conjunto de datos. Se basa en el conocimiento previo o "a priori" de los conjuntos frecuentes, esto sirve para reducir el espacio de búsqueda y aumentar la eficiencia. (Wikipedia, Algoritmo apriori)

Eclat (Equivalence Class Transformation):

- Según la definición de clase de equivalencia, los k-itemsets se pueden particionar, agrupándolos por prefijos iguales de tamaño k - 1.

Algoritmo FP-Growth (Lucas, 2015)**:**

- Permite de detección de set de ítems frecuentes sin la necesidad de generar un set de ítems candidato; utilizando una estructura de datos alternativa llamada *FP-Tree* (árbol de patrones frecuentes), en donde se almacena información sobre el conjunto de ítems frecuentes. La contruccion de la estructura F-Tree representa los ítems frecuentes encontrados.
- Resultado igual a Apriori.

2.5 Estado del arte

Con el fin de robustecer el estudio de los antecedentes del presente trabajo, se seleccionó un conjunto de *papers* que representan las áreas de interés de la investigación, donde se realizó un análisis y evaluación de los mismos.

Uno de los trabajos más relacionados con la presente investigación es el titulado "*Mining Performance Regression Testing Repositories for Automated Performance Analysis* (King Chun Foo, 2010)". En dicho *paper* los autores presentan una solución automatizada para la detección de problemas potenciales en el rendimiento de un sistema, al aplicar pruebas de regresión. La propuesta compara datos de las últimas pruebas con correlaciones de métricas de rendimiento pre-computadas, extraídas de un repositorio de métricas de rendimiento.

Los autores atacan o se centran en los resultados de las pruebas de regresión, los cuales también llaman "degradación de rendimiento"; en este caso se refieren a comparaciones en el tiempo entre pruebas de regresión de *releases* anteriores con las nuevas pruebas de regresión, con el fin de detectar si hubo o no, un deterioro en el rendimiento de la aplicación.

Un concepto interesante planteado en su investigación es que crean y hacen uso de un repositorio con los resultados de las pruebas de rendimiento, con el fin de ser analizados en el tiempo y no solo los datos o los resultados arrojados de las pruebas de rendimiento actuales.

Los autores utilizan dos conceptos de *datamining:* las reglas de asociación y los sets de *ítems* frecuentes, en los cuales utilizan el algoritmo Apriori para la definición de las reglas.

Dicho estudio tiene objetivos similares a los planteados en esta investigación, como la utilización de reglas de asociación y la obtención de métricas de los componentes de hardware, por ejemplo el CPU y la memoria. Sin embargo, hay diferencias que se pueden mencionar.

En efecto, la presente investigación pretende atacar uno de los problemas que la mayoría de aplicaciones y organizaciones están enfrentando: la gran cantidad de información o datos que son generados y procesados hoy en día.

En la actualidad el pre procesamiento de los datos es un gran reto, ya que generalmente dichos datos son generados por diferentes fuentes y en diferentes formatos; por lo que se habla de datos masivos con estructuras y formatos complejos.

Por medio de ese trabajo se busca la mejora del proceso de análisis y evaluación de las pruebas de rendimiento de software en ambientes de *Big Data,* el cual permita analizar este tipo de datos masivos y complejos en menor cantidad de tiempo. Además, se propone una manera de estandarizar estos procesos por medio de este tipo de herramientas.

La definición de estándares como el formato de los archivos por recolectar y el procesamiento de los datos, ayudan a la identificación de escenarios que puedan perjudicar el rendimiento de los sistemas.

Asimismo, se utiliza la misma técnica de minería de datos para la obtención de reglas de asociación; sin embargo, la diferencia radica en que la forma de correlacionar los datos es distinta, ya que también se persigue obtener una relación entre el rendimiento de los componentes de hardware, como con los escenarios de prueba aplicados. De esta manera, también será más fácil para los miembros de los equipos de desarrollo, identificar cuales módulos o partes de la aplicación están afectando el desempeño del sistema y deben ser mejorados.

En otro estudio, lo autores de *"WAP 5: Black-box Performance Debugging for Wide-Area Systems"* (Patrick Reynolds, 2006) proponen la automatización de la detección de "cuellos de botella" o *delays* entre los nodos de un sistema distribuido.

Dicha propuesta analiza "*trace messages*" (rastreo de mensajes) de la comunicación entre los nodos del sistema, para identificar cuales de ellos tienen un *delay* o demora significativa a la hora de responder ante una determinada solicitud.

Los investigadores de dicho estudio proponen una solución para manejar aplicaciones, en las que no se conoce o no se tiene disponible el código fuente, donde el formato de los mensajes es desconocido y en general, no se tiene información sobre el diseño de la aplicación.

Es un modelo basado en la causalidad de mensajes entre los componentes de un sistema distribuido. En la investigación, los autores desarrollaron algoritmos y una librería para analizar el tráfico de los mensajes creados durante la ejecución de un sistema.

El trabajo tiene una propuesta interesante, pero se considera más un complemento al área de análisis del comportamiento de las aplicaciones y por ende al análisis de las pruebas de desempeño.

Para efectos de este proyecto, uno de los supuestos es que la aplicación, su arquitectura y el código son conocidos por el equipo de trabajo, ya que es necesario e indispensable para la identificación de los escenarios de prueba por evaluar. Otra razón por la cual es necesario conocer la aplicación y su arquitectura es que se debe definir qué es aceptable y qué no lo es a la hora de analizar los resultados obtenidos.

Uno de los aspectos importantes de la presente investigación es correlacionar las métricas de software y hardware propuestas. Así, es interesante la propuesta planteada en el artículo *"System Monitoring with Metric-Correlation Models: Problems and Solutions"* (Miao Jiang, 2009).

El estudio de los autores consiste en capturar la relación entre métricas por medio de varianza constante y no constante, la cual pueden modelar. Se utilizaron pruebas estadísticas para la identificación de las relaciones, las cuales pueden ser afectadas por

varianzas no constantes. Por medio de un sistema automatizado de monitoreo y del modelo propuesto pueden capturar el estado, comportamiento y rendimiento del sistema.

La solución propuesta es una opción existente para lograr correlacionar las métricas; sin embargo, se considera fuera del alcance, ya que como se ha mencionado anteriormente parte de este proyecto es la utilización de técnicas de minería de datos como las reglas de asociación; con el fin de relacionar los datos y de esa forma obtener información importante para decidir si un sistema informático está listo para funcionar en ambientes de producción. La utilización de las reglas de asociación para mejorar los análisis de las pruebas de desempeño soporta la hipótesis propuesta en esta investigación, por lo que se considera una opción interesante de explorar en futuros trabajos relacionados a este.

Finalmente, se encontraron trabajos relacionados a la mejora del funcionamiento del algoritmo Apriori, entre ellos se encuentran las investigaciones realizadas en el trabajo *"The Optimization and Improvement of the Apriori Algorithm"* (Yiwu Xie, 2008).

Los autores realizan un análisis del algoritmo Apriori y descubren aspectos que afectan la eficiencia del algoritmo. Los aspectos que mencionan son los siguientes:

- El escaneo frecuente de la base de datos.
- La gran escala del conjunto de elementos candidatos.

Ellos proponen un nuevo algoritmo al que llaman ***IApriori,*** el cual reduce los tiempos de escaneo de la base de datos, además de optimizar el proceso de unión del conjunto de elementos frecuentes generados.

Esto ayuda a la reducción del tamaño del conjunto de elementos candidatos lo que resulta en una mejora en la eficiencia del algoritmo.

Existen otras mejoras similares al algoritmo Apriori que pueden ser consultadas en (Wang, 2010); (Lei Ji, 2006) (Weixiao LIU).

Otro *paper* que plantea una mejora al algoritmo es el titulado *"A Probabilistic Approach to Apriori Algorithm"* (Vaibhav Sharma, 2010). Este *paper* propone, de igual forma una mejora al algoritmo Apriori por medio de conceptos de probabilidad para la generación de

sets de datos frecuentes en una base de datos transaccional. El algoritmo propuesto mejora al original para bases de datos muy grandes, sin perder una sola regla. La mejora busca disminuir el número de candidatos no exitosos generados por el algoritmo, ya que más adelante afectan el mínimo soporte utilizado por el mismo.

Utiliza el concepto de medianas recursivas para calcular la dispersión en la lista de transacciones de cada set de datos. Estas son implementadas en el algoritmo como un árbol de búsqueda invertido ***V-Median***.

Las medianas recursivas se utilizan para calcular el número máximo de transacciones comunes. Los autores tratan de presentar un mecanismo probabilístico eficiente para descubrir conjuntos de elementos frecuentes.

El enfoque probabilístico presentado resulta interesante como una de las posibles mejoras al algoritmo Apriori.

Por último, los investigadores de "*Applying Correlation Threshold on Apriori Algorithm"* (Anand H.S., 2013) plantean una mejora a la eficiencia del algoritmo, evaluando el número de los elementos candidatos generados. El esquema propuesto genera un mayor conjunto de elementos frecuentes en menor tiempo. Se logró determinar que el tiempo se redujo al orden de O(n).

El tiempo se reduce debido al número de escaneos a la base de datos. El escaneo dependía de la longitud del set de datos de elementos frecuentes el cual fue suplantado por la introducción de un arreglo probabilístico. Los resultados confirmaron que la modificación al algoritmo fue positiva, pues mejoró la eficiencia y redujo el tiempo.

El enfoque presentado al incluir este nuevo factor de correlación, según los resultados de los autores, mejora significativamente la eficiencia o tiempo invertido por el algoritmo. La investigación propone una ampliación del algoritmo Apriori. La solución incluye un nuevo atributo o factor de correlación al que los autores llaman ***"umbral"***.

Es interesante la comparación de los diferentes métodos expuestos para mejorar el algoritmo. Sin embargo, en la presente investigación, y como se ha tocado el tema previamente, los resultados de las pruebas de rendimiento pueden llegar a genenerar

grandes cantidades de información dependiendo del alcance del sistema informático, la arquitectura y el período seleccionado para la ejecución de las mismas.

Por lo tanto, se podría hablar de Petabytes de datos en formatos distintos y de gran complejidad, por lo que para abordar este problema se utilizaron técnicas y herramientas de *Big Data.* Por ello, la evaluación de los métodos y técnicas expuestas en otros estudios para mejorar el algoritmo quedan fuera del alcance de la investigación, aunque puede resultar interesante evaluarlo en investigaciones posteriores a este trabajo.

Se puede notar, que existen varios estudios relacionados al análisis de las pruebas de rendimiento y la generación de reglas de asociación; no obstante no se identificó un estudio referente a la identificación de patrones mediante el uso de reglas de asociación. La correlación de los resultados de las métricas de los componentes de hardware con las pruebas funcionales es el principal objetivo que persigue esta investigación.

Otro aspecto importante, es que la utilización de herramientas de *Big Data* es un elemento clave en el desarrollo de esta investigación, ya que el procesamiento de los datos se ha identificado como uno de los retos a resolver.

Los artículos sobre las mejoras del algoritmo Apriori, se considerán posterioirmente en el capítulo de reomendaciones de este trabajo.

CAPÍTULO 3 Propuesta metodológica

Con el propósito de validar la hipótesis planteada en la presente investigación, se procedió a desarrollar una metodología, la cual se describe en el este capítulo.

Para la definición del modelo, se toma en cuenta uno de los procesos estandarizados, definido en la actualidad para minería de datos, llamado CRIPS-DM del inglés *"Cross Industry Standard Process for Data Mining"* (Wikipedia, Cross Industry Standard Process for Data Mining).

El proceso define varios pasos para analizar un problema de minería de datos. Entre ellos, se tiene: que entender el negocio, comprender los datos por evaluar, preparar los datos, el modelado de la solución, la evaluación y finalmente el *deployment.*

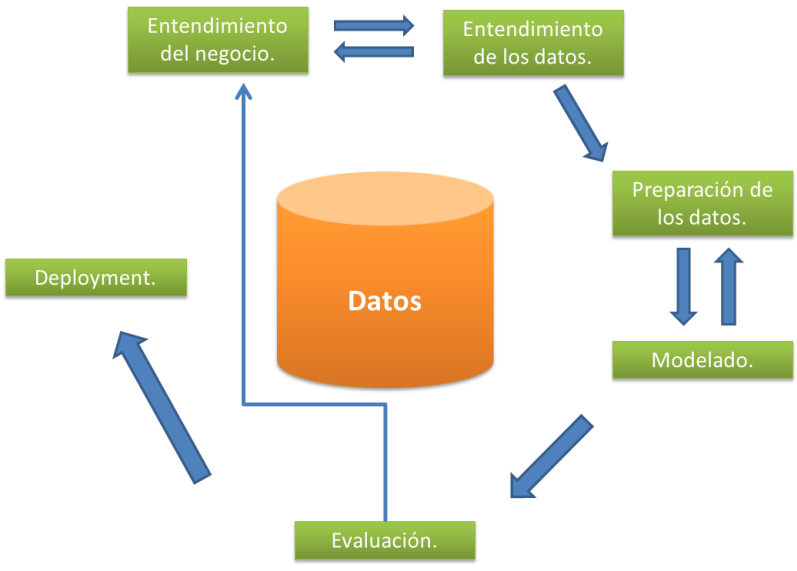

Figura 4. Modelo CRISP-D.

Fuente: Traducida de (Wikipedia, Cross Industry Standard Process for Data Mining)

Al modelo propuesto se le llamó APRA "*Análisis de pruebas de rendimiento automático*". En dicho modelo se definen una serie de pasos que deben cumplirse para llegar al resultado deseado.

En la figura 5, se muestra de forma general un esquema de APRA.

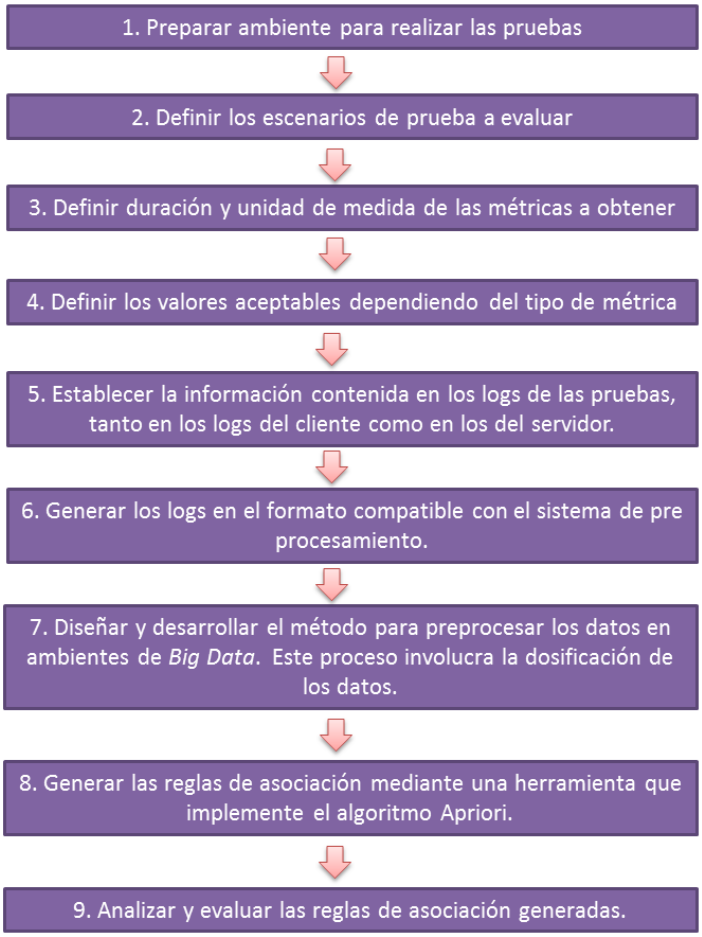

Figura 5. Pasos a alto nivel del modelo APRA.

(Fuente: Creación propia).

El modelo considera una estandarización en la manera como se captura la información de las pruebas; ya que, la delimitación de los datos en los archivos de *logs* debe ser por medio de comas, la herramienta utilizada para el procesamiento debe ser del *suite* de herramientas disponibles en las distribuciones de Hadoop y, finalmente, la implementación del algoritmo Apriori para la generación de las reglas de asociación.

Es muy importante definir tanto los escenarios de prueba, como los parámetros capturados en los *logs*. Por otro lado, las unidades de medición de las métricas de hardware y software evaluados son de suma importancia, ya que el sistema depende del análisis o la aprobación para mover a ambientes de producción.

La definición de lo anterior es fundamental para lograr los resultados esperados al final del procesamiento de los resultados de las pruebas de rendimiento del sistema.

La evaluación de las reglas de asociación será un proceso manual, que los integrantes del equipo de desarrollo deberán discutir y evaluar para tomar la decisión final. Cabe mencionar que la presente investigación no pretende implementar un mecanismo para visualizar los resultados, aunque es un aspecto que podría ser desarrollado en un trabajo o estudio posterior.

3.1 Modelo propuesto.

El modelo consta de 14 pasos detallados a seguir, en el cual es requisito cumplir con cada paso para continuar con el sucesivo. La tabla 1 muestra los pasos que se deben considerar.

Tabla 1. Pasos del modelo APRA.

MODELO APRA
1. Contar con los pre-requisitos antes de iniciar con el modelo. 1.1 Contar con el ambiente para realizar las pruebas. 1.2 Contar con *cluster* con Hadoop configurado y listo para llevar a cabo el pre procesamiento de los datos. 1.3 Contar con una herramienta de minería de datos para aplicar el algoritmo Apriori compatible con el modelo propuesto.
2. Definir los escenarios y la duración de las pruebas de la aplicación.
3. Definir las unidades de medidas y valores aceptables para cada uno de los elementos a medir en el modelo.
4. Incluir en los logs del cliente la información requerida por el modelo. • Momento en que ocurrió la prueba. • Tipo de prueba o escenario a probar.

MODELO APRA
• Tiempo de respuesta.
5. Incluir en los logs del servidor la información requerida por el modelo. • Momento cuando se recopila el estado de los componentes de hardware del servidor. • Porcentaje de procesador utilizado. • Cantidad de memoria RAM disponible. • Tamaño de las solicitudes de I/O a disco. • Transferencia en bytes de la interface de red.
6. Generar los logs del cliente y servidor en un formato compatible con el sistema que se utiliza.
7. Copiar los archivos de logs generados por las pruebas al sistema local de archivos de la computadora con Linux donde está configurado Hadoop.
8. Cargar los archivos al HDFS (*Hadoop Distributed File System*).
9. Crear la base de datos correspondiente en Hadoop.
10. Ejecutar el script .hql desarrollado en esta investigación para limpiar, dosificar y preparar los datos.
11. Generar el archivo plano desde la línea de comandos de Hadoop.
12. Crear el archivo compatible con el sistema elegido (basado en el archivo final generado en el paso anterior) para la ejecución del algoritmo Apriori.
13. Generar las reglas de asociación por medio de un sistema que implemente el algoritmo Apriori que sea compatible con el modelo propuesto.
14. Analizar las reglas de asociación obtenidas y tomar las decisiones correspondientes de los resultados de las pruebas.

3.2 Descripción de los pasos del modelo propuesto (APRA)

1. Contar con los siguientes pre-requisitos antes de iniciar con el modelo.

Antes de iniciar con el modelo para el análisis de las pruebas de desempeño es requerido contar con lo siguiente:

a) Contar con el ambiente para realizar las pruebas.

Es de suma importancia contar con las computadoras (servidores, laptops o computadoras de escritorio) en donde se aplicarán las pruebas, simulando ambientes de producción.

b) Contar con *cluster* con Hadoop configurado y listo para llevar a cabo el pre-procesamiento de los datos.

El modelo cuenta con una serie de pasos para pre procesar los datos y generar un *output* homogéneo y estructurado de los datos recolectados, por lo que requiere de un *cluster* con Hadoop instalado y configurado.

El *output* será generado en pasos posteriores del modelo.

c) Contar con una herramienta de minería de datos para aplicar el algoritmo Apriori compatible con el modelo propuesto.

Finalmente, se necesita de una herramienta con la implementación del Algoritmo Apriori, esta será la encargada de generar las reglas de asociación.

2. Definir los escenarios y la duración de las pruebas de la aplicación.

Antes de iniciar con las pruebas el equipo de desarrollo debe identificar el conjunto de pruebas con los escenarios o funcionalidades por probar. Una vez definidas las pruebas se deben evaluar y analizar durante cuánto tiempo deben ser ejecutadas las pruebas.

3. Definir las unidades de medidas y valores aceptables para cada uno de los elementos a medir en el modelo.

Después de definir el conjunto de pruebas o escenarios, se debe determinar la unidad de medida para cada elemento. Al referirse a elementos se habla de:

- El tiempo de respuesta para cada escenario de prueba.
- Porcentaje de procesador utilizado.
- Cantidad de memoria RAM disponible.
- Tamaño de las solicitudes de I/O a disco.
- Transferencia en bytes de la interface de red.

Por ejemplo, actualmente la memoria RAM es dada en GB, por lo que dependiendo del detalle que se quiera obtener variaría; podrían ser MB o KB por ejemplo. De igual manera deben definirse las unidades de medida para el resto de los elementos.

Cabe mencionar que el tipo y cantidad de *performance counters* o componentes de hardware por medir, depende de lo que se considere necesario y requerido a medir en las pruebas. El script implementado en este trabajo y la manera de procesar los datos se puede ajustar dependiendo de ello. Para efectos de esta investigación se utilizarán los antes mencionados.

Finalmente, en este paso se debe establecer cuales son los valores aceptables para cada medida, esto con el fin de decidir en qué escenarios la aplicación no está teniendo el comportamiento esperado.

4. **Incluir en los logs del cliente la información requerida por el modelo.**

El modelo propuesto requiere que el archivo de *log* generado por el cliente cuente al menos con la siguiente información:

- **Momento en que ocurrió la prueba:**

  Este valor debe ser almacenado con el siguiente formato:

  - mm/dd/yyyy hh:mm:ss

- **Tipo de prueba o escenario a probar:**

  El tipo de prueba corresponde a los escenarios de prueba identificados en pasos anteriores.

  Es importante identificar o clasificar la prueba, esto ayudará a la detección de los escenarios que están impactando el desempeño de la aplicación; si es que se llega a encontrar algún patrón negativo en las reglas generadas.

- **Tiempo de respuesta:**

  El tiempo de respuesta de la prueba muestra cuánto tardó la aplicación en finalizar la solicitud para ejecutar un determinado proceso, funcionalidad o escenario de prueba. Este dato también es fundamental para determinar el desempeño del sistema.

5. **Incluir en los logs del servidor la información requerida por el modelo.**

El modelo propuesto requiere que el archivo de log generado por el servidor considere al menos con la siguiente información:

- **Momento cuando se recopila el estado de los componentes de hardware del servidor:**
  Este valor debe ser almacenado con el siguiente formato:
  - mm/dd/yyyy hh:mm:ss
- **Porcentaje de procesador utilizado:**
  Este valor representa el porcentaje de procesador que está siendo utilizado en un momento determinado
- **Cantidad de memoria RAM disponible:**
  Valor que muestra la cantidad de memoria disponible en un momento determinado. Generalmente medido en GB.
- **Tamaño de las solicitudes de I/O a disco:**
  Muestra el tamaño promedio de las solicitudes de disco (tamaño de I/O) para un momento determinado.
- **Transferencia en bytes de la interface de red:**
  Muestra la velocidad a la que la interface de red procesa los datos en un determinado momento.

Es importante notar que la frecuencia de escritura en este log también debe ser definida antes de iniciar con las pruebas, ya que durante las pruebas estos logs funcionan como un sistema de monitoreo de los componentes de hardware. En pasos posteriores del modelo se aplicarán mecanismos para correlacionar los datos entre sí.

6. **Generar los logs del cliente y servidor en un formato compatible con el sistema que se utiliza.**

   Se debe considerar el formato utilizado para escribir la información en los logs. Un ejemplo puede ser un archivo plano, en el cual cada valor es separado por comas.

7. **Copiar los archivos de logs generados por las pruebas al sistema local de archivos de la computadora con Linux donde está configurado Hadoop.**

   Los archivos con los resultados de las pruebas deben ser copiados al sistema de archivos local en el nuevo folder creado.

8. **Cargar los archivos al HDFS (*Hadoop Distributed File System*).**

   Como buena práctica considerar la creación de un folder nuevo en donde se copiarán los archivos a cargar en el sistema. Esto se lleva a cabo mediante la línea de comandos de Linux.

   Para crear un nuevo folder desde la consola de Linux se debe ejecutar el siguiente comando:

   - hadoop fs -mkdir [Nuevo Path],

   en donde "Nuevo Path" especifica la ubicación en el sistema de archivos de Hadoop.

   Después de crear el folder en el HDFS se deben cargar los archivos del cliente y el servidor ejecutando el siguiente comando:

   - hadoop fs -put [/Folder Fuente/ClientLog.csv] [/Nuevo Path]

   El primer parámetro indica la fuente de donde se tomarán los archivos y el segundo indica el destino al que serán copiados los mismos.

9. **Crear la base de datos correspondiente en Hadoop.**

   En este paso es necesario conectarse a *Hive* y crear la base de datos donde se almacenarán los datos inicialmente basados en los archivos generados. Este paso es de suma importancia para que posteriormente se realice el proceso de dosificación, pre

procesamiento y preparación de los datos. La conexión a Hive se realizará por medio de beeline (Atlassian, 2015):

- beeline -u jdbc:hive2://[Host]:[Port]/[DB] -n [Usuario] -d [Driver]

Ejemplo:

- beeline -u jdbc:hive2://quickstart:10000/default -n cloudera -d org.apache.hive.jdbc.HiveDriver

Para crear la base datos ejecutar el siguiente comando:

- Create database [Nueva Base de Datos];

10. Ejecutar el script .hql desarrollado en esta investigación para limpiar, dosificar y preparar los datos.

Se implementó un HQL script para el pre procesamiento de los datos. La ejecución de este script tiene como objetivo generar un solo archivo basado en el archivo del cliente y del servidor.

El script se encarga de limpiar, podar, dosificar, clasificar y relacionar los registros de un archivo con el otro. Esto con el fin de obtener un solo archivo con el formato correcto para aplicar finalmente el algoritmo Apriori.

Otro aspecto importante a mencionar, es que el script también se encarga de clasificar los datos, asignándoles un valor dentro de un conjunto de valores discretos. La clasificación de los datos es muy importante, ya que ayuda al algoritmo Apriori a identificar conjuntos de datos frecuentes que posteriormente ayudarán a identificar patrones. El script debe ser ajustado de acuerdo a los tipos de clasificación definidos para las pruebas y el conjunto de datos obtenido. Un ejemplo de clasificación puede ser “Memoria disponible entre 7 GB y 8 GB”. La clasificación utilizada para efectos de esta investigación está detalladas en las Tablas 5 y 6 de este documento.

Para ver el código del script ir al Anexo A “Formateo y limpia de los datos por medio de un HQL Script”.

11. Generar el archivo plano desde la línea de comandos de Hadoop.

La generación del archivo final debe ser desde la consola de *Hive*. Para crearlo se deben ejecutar las siguientes instrucciones:

- Conectarse a Hive.
- Indicar cuál base de datos se utilizará.
  - use tesistec;
  - En este caso se definió “tesistec” como el nombre de la base de datos a utilizar.
- Indicar que se incluyan los encabezados mediante la siguiente instrucción.
  - set hive.cli.print.header=true;
- Crear el archivo basado en la tabla final.
  - INSERT OVERWRITE LOCAL DIRECTORY [Folder Destino] ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' select * from finalLog;

12. Crear el archivo compatible con el sistema elegido (basado en el archivo final generado en el paso anterior) para la ejecución del algoritmo Apriori.

Se formatea el archivo para que sea compatible con el sistema elegido para la ejecución del algoritmo Apriori de acuerdo a las especificaciones requeridas.

13. Generar las reglas de asociación por medio de un sistema que implemente el algoritmo Apriori que sea compatible con el modelo propuesto.

En este paso del modelo se aplica y ejecuta la implementación de Apriori seleccionada. Como *input* a este paso se debe considerar el archivo creado y formateado en el paso anterior.

14. Analizar las reglas de asociación obtenidas y tomar las decisiones correspondientes de los resultados de las pruebas.

Finalmente, al obtener las reglas de asociación generadas se procede con la evaluación y el análisis de las mismas; con el fin de detectar patrones que puedan impactar de manera negativa el rendimiento o desempeño de la aplicación.

Es importante mencionar que en este paso el analista de las reglas obtenidas deberá generar los resultados por medio del uso de diferentes niveles de confianza (soporte) para determinar cuál es el adecuado para realizar el análisis.

Una vez identificado el nivel de confianza para los datos obtenidos de las pruebas de rendimiento se podrán evaluar las reglas e identificar los riesgos potenciales en la aplicación, si es que se detectara alguno.

Dependiendo del resultado se tomarán las acciones correspondientes, ya sea para aprobar el pase a producción o solucionar los problemas de desempeño detectados.

CAPÍTULO 4 Desarrollo de la propuesta

Una vez definido el modelo, se procede a validar la propuesta metodológica planteada, en la que se definen las características de los componentes de hardware y de software involucrados en las pruebas de desempeño de la aplicación.

Las características de hardware requieren contemplar las especificaciones de los dispositivos o componentes contenidos en el servidor y computadora cliente. En el caso del software, se refiere a las características de las pruebas, como los escenarios o tipos de pruebas por evaluar, la frecuencia con que serán aplicadas o ejecutadas, y el desempeño esperado para cada una de las pruebas definidas.

Asimismo, se presentarán de manera detallada los pasos llevados a cabo para el pre-procesamiento de los datos. Este paso es de suma importancia en el desarrollo del modelo propuesto, ya que, conlleva a uno de los principales objetivos de la investigación, como es el desarrollar una solución de *Big Data* para el procesamiento y preparación de los datos antes de aplicar el algoritmo Apriori.

Como se ha mencionado anteriormente, el modelo propuesto debe ser escalable y permitir el procesamiento de datos masivos y complejos; por lo que el desarrollo de este proceso, particularmente, es de suma importancia para la propuesta planteada.

Finalmente, se detallará el paso de la generación de las reglas de asociación por medio de una herramienta con la implementación de Apriori, que permita la generación de las mismas para un análisis posterior.

4.1 Especificaciones del Cliente y Servidor

Para el análisis y evaluación de las pruebas se debe de utilizar un ambiente de prueba similar a las características del ambiente de producción.

A continuación, en la tabla 2, se muestran las características del ambiente de prueba utilizado para la ejecución de los distintos escenarios definidos para la aplicación de alta disponibilidad seleccionada. Basado en este ambiente se recopilan los datos de las pruebas de rendimiento.

Tabla 2. Descripción de las características de hardware y Sistema operativo del cliente y servidor

Capa	Descripción	Producción	Ambiente de Simulación
Cliente	Clientes concurrentes estimados	50	50
	Hardware del Cliente	Estándar IT laptop	Máquina Virtual CPU: Intel® Xeon® 2.67GHz(2 procesadores). Memoria RAM: 2 GB.
	Sistema Operativo	Windows 7	Windows Server 2008 R2 Edition.
Server. Servicio de Windows	Hardware de la capa del Servicio	H/W Modelo: BL-460C G6 Memoria RAM: 12 GB.	H/W Modelo: DL-380 G4 Memoria RAM: 12 GB.
	Sistema Operativo	Windows Server 2008 R2 Edition. SP1.	Windows Server 2008 R2 Edition. SP1.

4.2 Descripción de las transacciones o pruebas, frecuencia y desempeño esperado

De acuerdo a las funcionalidades más utilizadas y más importantes de la aplicación, se identificaron escenarios de prueba o transacciones. A cada una de las transacciones se le estimó la frecuencia con la que podrían ser llamadas y cuántas solicitudes concurrentes podrían presentarse durante el período de la prueba. Para poder definir esto, se requiere del juicio experto del equipo de trabajo.

La tabla 3 muestra la definición de los escenarios de prueba con su frecuencia de ejecución y tiempo de respuesta esperado.

Tabla 3. Definición de las características de las pruebas a ejecutar.

Nombre de la transacción o prueba	Frecuencia	Solicitudes concurrentes	Tiempo de respuesta esperado en segundos
GetApplicationModules	*50*	*1-50*	*< 1*
GetClientPendingServerEvents	*280,000*	*1-50*	*< 1*
GetCurrentWorker	*50*	*1-50*	*< 1*
GetEvent	*250*	*1-50*	*< 2*
GetEventContainmentSummaryData	*50*	*1-5*	*< 1*
GetEventHeader	*1000*	*1-50*	*< 1*
GetNotificationDetails	*2000*	*1-50*	*< 1*
GetNotificationHistoryCurrentUser	*50*	*1-50*	*< 1*
GetTimeZones	*50*	*1-50*	*< 1*
GetUiRequestConfiguration	*50*	*1-50*	*< 1*
GetUserPermissions	*50*	*1-50*	*< 1*
ProcessTraceResults	*60*	*1-6*	*< 60*
RegisterForServerEvents	*50*	*1-50*	*< 1*

4.3 Pre-procesamiento de los datos

4.3.1 Análisis de los datos

Los resultados de las pruebas son almacenados en archivos planos, en los cuales cada valor es separado por una coma. Dependiendo de la arquitectura de la aplicación por probar, la complejidad de los escenarios y el período de tiempo de la corrida de las pruebas, así será la cantidad de información generada. Debido a esto es necesario tomar en cuenta que para la preparación y la poda de los datos tendremos que procesar probablemente grandes cantidades de información.

La solución propuesta debe ser escalable en el tiempo y entre los diferentes tipos de aplicaciones, es por esta razón que es importante tomar en cuenta la utilización de herramientas para *Big Data"*.

Las pruebas, como se ha mencionado anteriormente, generan dos tipos de archivos de logs:

1. **En el cliente:**

    - Son los resultados y tiempo de respuesta de cada llamado o simulación de los escenarios de prueba.

2. **En el servidor:**

    - Resultado de los *performance counters* del servidor.

Particularmente, en este caso contamos con 2 logs de pruebas, para poderlos relacionar es necesario podar o dosificar los datos y procesarlos de tal manera que el conjunto de datos analizado pueda ser relacionado.

Para esto, se han identificado una serie de pasos a seguir:

1. La manera de relacionar los registros de un archivo de log con el otro será a través de la columna o dato llamado "*timestamp*". Es muy importante tomar en cuenta que el tiempo en el servidor y la computadora cliente deben estar sincronizados.

    El *tiemestamp* es almacenado en el siguiente formato:

    - dd/mm/yyyy hh:mm:ss, por ejemplo: 10/8/2012 4:08:42 PM

2. El tiempo de ejecución de la prueba es variable, debido a esto y para poder relacionar los datos es necesario estandarizar la forma en que son almacenados.

    En este caso se agruparán los registros por dd/mm/yyyy hh:mm, en donde se calcula el promedio del tiempo de respuesta en segundos, generando un solo registro para el grupo seleccionado. Con base en la cantidad de datos y la frecuencia de escritura en el archivo de log, la manera de agrupar los datos puede variar. En este caso, y como se mencionó anteriormente, se hará por medio de segundos.

3. Para los logs creados en el servidor se llevará a cabo el mismo proceso para el *timestamp* en el archivo de logs correspondiente. Así, se obtendrá un promedio para

tiempo de procesador, memoria utilizada, bytes transferidos por la red, y escritura/lectura a disco.

De esta manera, podremos relacionar los datos, en donde el ID de la transacción será el *timestamp* de los dos archivos de logs.

4. Por otra parte, la propuesta para comparar y evaluar el tiempo de respuesta y la cantidad o porcentaje utilizado del dispositivo de hardware será por medio de rangos. Un ejemplo de ello se muestra en la tabla 4.

Tabla 4. Ejemplo de clasificación de prueba por tiempo de respuesta

Timestamp	Operation	Secs	Classification
10/8/2012 16:08:42	GetClientPendingServerEvents	23.49	20-30
10/8/2012 16:13:58	GetClientPendingServerEvents	21.604	20-30
10/8/2012 17:15:12	GetClientPendingServerEvents	24.138	20-30
10/8/2012 17:20:24	GetClientPendingServerEvents	21.71	20-30
10/8/2012 17:41:24	GetClientPendingServerEvents	21.326	20-30
10/8/2012 18:00:00	GetClientPendingServerEvents	2.18	20-30
10/8/2012 22:20:19	GetClientPendingServerEvents	25.74	20-30
10/8/2012 22:25:30	GetClientPendingServerEvents	21.889	20-30
10/8/2012 22:46:27	GetClientPendingServerEvents	21.763	20-30
10/9/2012 3:22:17	GetClientPendingServerEvents	24.639	20-30
10/9/2012 3:27:38	GetClientPendingServerEvents	21.643	20-30
10/9/2012 3:48:14	GetClientPendingServerEvents	21.389	20-30

4.3.2 Pre-procesamiento por medio de la herramienta de *Big Data* seleccionada

La herramienta a utilizar para procesar *Big Data,* es ***Hive*** **de Hadoop,** la cual se usa para la preparación y dosificación de los datos. El paso de pre-procesamiento de los datos es muy importante, ya que este será el insumo para la generación de las reglas de asociación.

Para el pre-procesamiento y preparación de los datos se realizaron los pasos detallados en la figura 6.

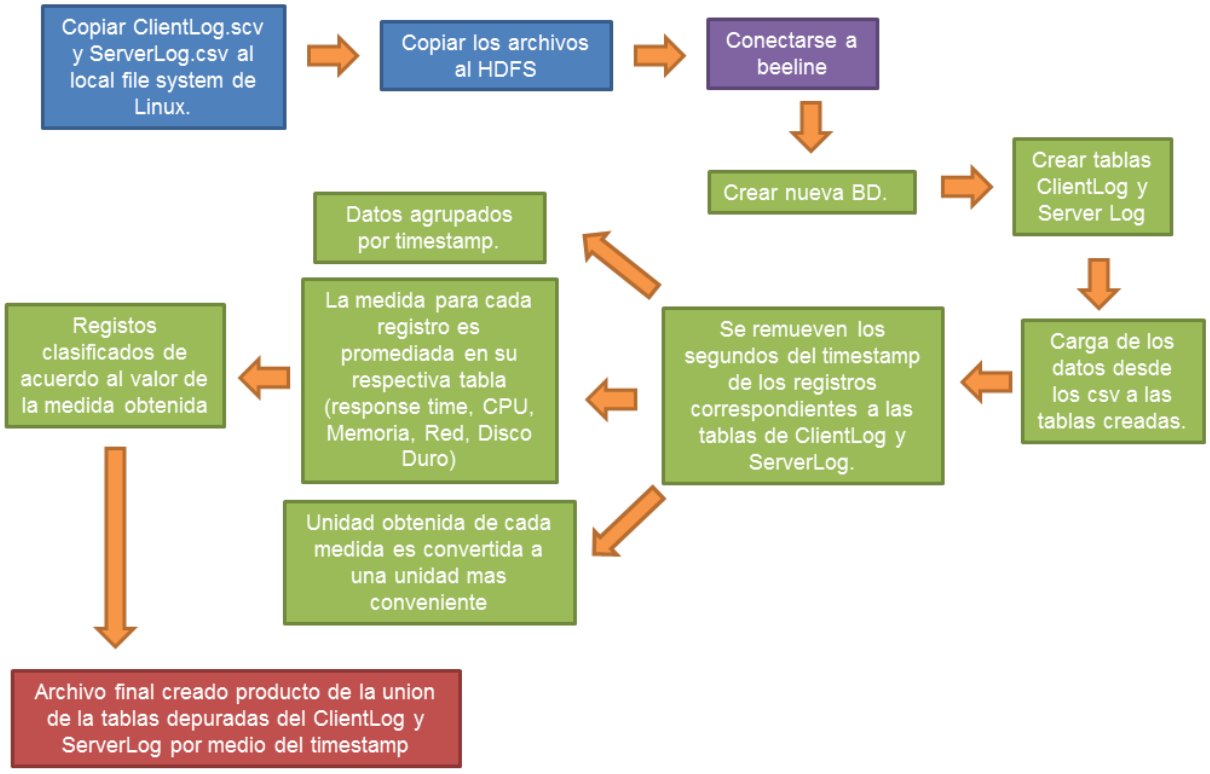

Figura 6. Descripción de implementación de pasos en el script de HQL para procesar, dosificar y preparar los datos.

(Fuente: Creación propia).

Los detalles sobre la implementación del script HQL se encuentran disponibles en el Anexo A del documento llamado “Formateo y limpia de los datos por medio de un HQL Script”.

4.3.3 Clasificación de los datos

Para el archivo generado en el lado del cliente se clasificaron los registros basados en el valor del *response time* o tiempo de respuesta obtenido de la siguiente manera:

Tabla 5. Clasificación del Log del Cliente.

Log del Cliente	
Response Time	• Operación o proceso a probar + _LessThan1sec. • Operación o proceso a probar + _Between1and2secs. • Operación o proceso a probar + _GreaterThan2secs.

Tabla 6. Clasificación del Log del Servidor.

Log del Servidor	
server_percentage_processor_time	• CPU_LessThan30. • CPU_Between30and60. • CPU_Between60and80.
memory_available_gb	• MemoryAvailableGB_LessThan1. • MemoryAvailableGB_Between1and2GB. • MemoryAvailableGB_Between2and3GB. • MemoryAvailableGB_Between3and4GB. • MemoryAvailableGB_Between4and5GB. • MemoryAvailableGB_Between5and6GB. • MemoryAvailableGB_Between6and7GB. • MemoryAvailableGB_Between7and8GB. • MemoryAvailableGB_Between8and9GB. • MemoryAvailableGB_Between9and10GB. • MemoryAvailableGB_GraterThan10.
network_KB_totalpersec	• NetworkKBPerSec_LessThan20KB. • NetworkKBPerSec_Between20and30KB. • NetworkKBPerSec_Between30and40KB. • NetworkKBPerSec_Between40and50KB. • NetworkKBPerSec_GreaterThan50KB.
hd_avg_disk_KB_transfer	• HDAvrgKBTransfer_LessThan10KB. • HDAvrgKBTransfer_Between10and20KB. • HDAvrgKBTransfer_Between20and30KB. • HDAvrgKBTransfer_Between30and40KB. • HDAvrgKBTransfer_Between40and50KB. • HDAvrgKBTransfer_Between50and60KB. • HDAvrgKBTransfer_Between60and70KB. • HDAvrgKBTransfer_Between70and80KB. • HDAvrgKBTransfer_Between80and90KB. • HDAvrgKBTransfer_Between90and100KB. • HDAvrgKBTransfer_GreaterThan100KB.

4.3.4 Generación de las reglas de asociación

4.3.4.1 Algoritmo y herramienta seleccionada

Como se ha mencionado, el algoritmo seleccionado para correlacionar los datos y generar las reglas de asociación es el algoritmo Apriori. Existen varias herramientas y formas donde se ha implementado este algoritmo. Para esta investigación se ha seleccionado una de ellas, con el fin de obtener las reglas de asociación.

Esta es una herramienta que implementa una colección de algoritmos de *machine learning* para tareas de minería de datos. Los algoritmos pueden ser ejecutados directamente desde la aplicación o llamados desde un programa escrito en java. (Waikato., 2015)

Contiene herramientas para el pre procesamiento de datos, clasificación, regresión, *clustering*, reglas de asociación y visualización. El formato del archivo a utilizar para la generación de las reglas de asociación en esta investigación será .arff.

CAPÍTULO 5 Validación de resultados

En este capítulo, se analizan los resultados obtenidos después de realizado el pre procesamiento de los datos con *Hive* de Hadoop, por medio del algoritmo .hql implementado, y la herramienta seleccionada que implementa Apriori para la generación de las reglas de asociación.

Por medio de las reglas de asociación, el equipo de desarrollo deberá ser capaz de analizarlas, evaluarlas; y de esta forma, tomar las decisiones correspondientes, ya sea para resolver cualquier problema detectado o aprobar la puesta en marcha en producción del sistema puesto a prueba.

5.1 Resultados obtenidos al procesar el archivo generado con Hive

5.1.1 Configuración de la herramienta que implementa Apriori

La aplicación ofrece ciertos parámetros que pueden ser modificados antes de ejecutar el algoritmo, estos parámetros son ajustables de acuerdo a la naturaleza de los datos.

Dichos parámetros son (la mayoría de las definiciones de los parámetros fueron tomadas del mismo sistema, los cuales se encuentran detallados en la tabla 7):

Tabla 7. Definición de parámetros utilizados.

Parámetro	Definición
Car	Define si el algoritmo extraerá las reglas de asociación de clase en lugar de las reglas de asociación (general).
Classindex	Índice del atributo de la clase. Si se establece en -1, el último atributo se toma como atributo de clase.
Delta	Determina el factor de disfunción del soporte. La disminución se llevara a cabo de manera iterativa hasta que se alcance el número de reglas deseadas.
lowerBoundMinSupport	Indicamos el límite inferior de cobertura requerido para aceptar un conjunto de ítems.
metricType	Establece el tipo de métrica a usar para clasificar la regla. • ***Confidence*:** Confianza de la regla.

Parámetro	Definición
minMetric	Indicamos la confianza mínima (u otras métricas dependiendo del criterio de ordenación) para mostrar una regla de asociación.
numRules	Indicamos el número de reglas que deseamos obtener.
outputItemSets	Si estuviera habilitado (True) los diferentes *itemsets* o ítems de datos generados son imprimidos en pantalla.
removeAllMissingCols	Remueve las columnas con valores faltantes.
significanceLevel	Nivel Significativo. Las reglas son probadas de acuerdo al nivel significativo definido. El valor por defecto no aplica las pruebas. (Extended WEKA including Ensembles of Hierarchically Nested Dichotomies, 2015)
upperBoundMinSupport	Indicamos el límite superior de cobertura requerido para aceptar un conjunto de ítems.
Verbose	Si está habilitado (True) la aplicación corre en modo "Verbose".

Para efectos de las pruebas y validación de resultados se alterararon los siguientes parámetros (para los demás se utilizaron los valores por defecto):

- minMetric.
- numRules.

5.1.2 Interpretación de las reglas de asociación generadas

Como se ha mencionó anteriormente, para la generación de las reglas de asociación se variaron los parámetros de *minMetric,* el cual hace referencia al nivel de confianza o soporte y el número de reglas deseadas llamado *numRules*.

Se realizaron varias pruebas con el fin de determinar cuál era el nivel de soporte adecuado para la generación de las reglas de asociación.

Las pruebas realizadas fueron las siguientes:

1. Delta = 0.05 (5%), minMetric = 0.9, numRules = 10.
2. Delta = 0.05 (5%), minMetric = 0.7, numRules = 10.
3. Delta = 0.05 (5%), minMetric = entre 0.2 y 0.6, numRules = 10.
4. Delta = 0.05 (5%), minMetric = 0.9, numRules = 50.
5. Delta = 0.05 (5%), minMetric = 0.9, numRules = 200.

5.1.2.1 Hallazgos de las reglas

- Se puede deducir que durante el período cuando las pruebas fueron realizadas, el servidor siempre tuvo una memoria disponible entre 7GB y 8GB, con una utilización del procesador menor al 30%.

- También se puede mencionar que el tamaño promedio de las solitudes I/O al disco fueron entre 10 KB y 20 KB para una utilización de procesador menor al 30%.

- Para una velocidad de la interface de Red mayor a 50Kbs la utilización de procesador fue siempre menor al 30%.

- Para la mayoría de las pruebas el tiempo de respuesta fue de menos de 1 segundo, obteniendo una utilización de CPU de menos del 30% y con un porcentaje mayor al 50% de memoria disponible en el servidor (disponibilidad de ente 7GB y 8GB de memoria).

- El siguiente es un ejemplo de las reglas importantes detectadas después del análisis; donde se puede concluir que en los datos se encontró una relación entre el porcentaje de procesador utilizado, la memoria disponible y el tamaño promedio de las solicitudes de I/O hechas al disco.

    - *server_percentage_processor_time = CPU_LessThan30*

      memory_available_gb = MemoryAvailableGB_Between7and8GB 4290 ==>
      hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 conf:(0.87)

- Cabe mencionar que al cambiar los parámetros del nivel de confianza entre un 20% y 60% las reglas obtenidas no variaron; por lo que se puede concluir que para obtener 10 reglas utilizando el mismo conjunto de datos, la calidad de la información no cambia; así, se puede afirmar que durante las pruebas el tiempo de respuesta y las mediciones obtenidas para cada dispositivo de hardware fueron constantes para estas pruebas en particular.

- Al aumentar el número de reglas de asociación por obtener y aplicar el algoritmo se puede observar que hay una serie de reglas interesantes que involucran el tipo de operación y clasificación de tiempo de respuestas obtenidas en las pruebas. Esas reglas también son significativas para determinar el comportamiento de la aplicación y el hardware del servidor en las pruebas aplicadas.

- Después de aplicar el proceso de generación de reglas utilizando diferentes niveles de confianza, se puede concluir que con una confianza mayor o igual al 85 % se pueden obtener las reglas necesarias para realizar el análisis; dejando de lado las que son insignificantes desde el punto de vista de la evaluación de las pruebas de rendimiento para la aplicación utilizada.

- La calidad de las reglas obtenidas utilizando un nivel de confianza del 87% es considerado muy bueno y definitivamente sirve para obtener resultados confiables en el análisis de las pruebas.

- Finalmente, las reglas obtenidas implican una evaluación y un análisis muy positivo, logrando que la puesta en producción de la aplicación evaluada sea aprobada.

- Para ver las reglas encontradas durante el análisis de esta investigación ver Anexos del B al F.

5.1.3 Reglas de asociación obtenidas por escenario de prueba

En este caso particular se crea un archivo de resultados por cada tipo de escenario de prueba; de esta manera, es posible conocer con más exactitud y precisión el comportamiento de la aplicación y del servidor de acuerdo a cada tipo de prueba. Los escenarios de prueba son los definidos anteriormente, para más detalles ver la tabla número 3 del presente documento.

Los archivos generados para la validación de estas pruebas fueron los siguientes:

1. StartUpClients.
2. GetEvent.
3. GetEventHeader.
4. GetEventSummaryData.
5. GetNotifications.
6. GetPendingServerEvents.
7. ProcessTraceResults.

Una vez generado el archivo correspondiente a cada prueba, se formatea de acuerdo al formato compatible por la aplicación cliente encargada de generar las reglas de asociación. Al final se obtendrán dos archivos por cada prueba. En este caso en particularmente será un archivo plano separado por comas y otro en formato arff.

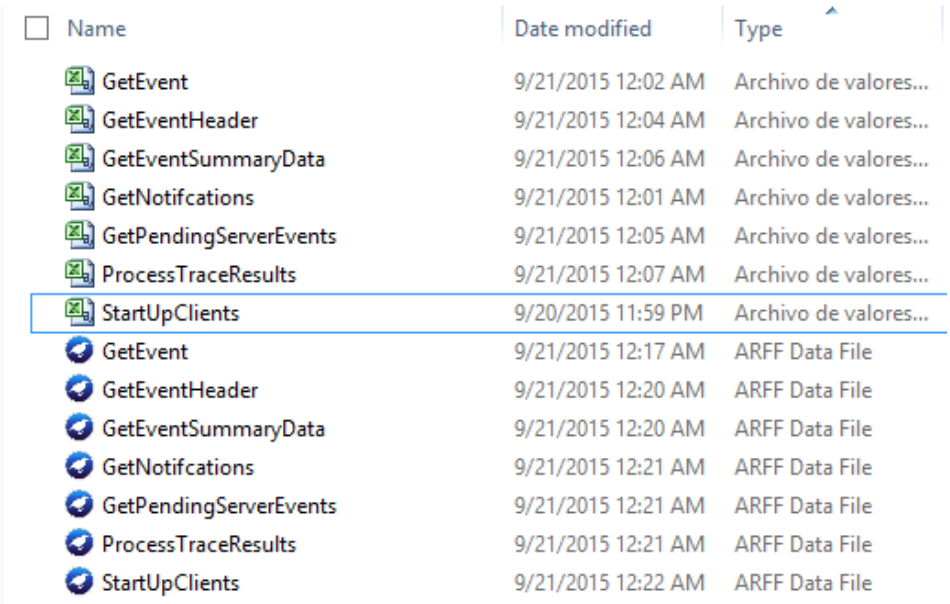

Figura 7. Archivos csv y arff generados por escenario de prueba.

5.1.4 Resultados de las reglas obtenidas por tipo de prueba o escenario.

Tabla 8. Reglas obtenidas por tipo de prueba o escenario

Escenario de Prueba	Observaciones
StartUpClients Confianza mayor al 85%.	**Delta = 0.05 (5%), minMetric = 0.8, numRules = 50** En este escenario se involucran los llamados a 2 tipos de procesos: • GetApplicationModules. • GetCurrentWorker. • GetTimeZones. • GetUIRequestConfiguration. • GetUserPermissions. • RegisterForServerEvents. A la hora de la ejecución de las pruebas cuando la memoria disponible osciló entre 7GB y 8GB, el porcentaje de utilización del CPU fue de menos del 30%. Para una velocidad de la interface de Red mayor a 50Kbs la utilización de procesador fue siempre menor al 30% y la memoria disponible osciló entre 7GB y 8GB. Por el otro lado, también se puede mencionar que el tamaño promedio de las solitudes I/O al disco fueron entre 10 KB y 20 KB para una utilización de procesador menor al 30%.
GetEvent Confianza mayor al 97%.	**Delta = 0.05 (5%), minMetric = 0.8, numRules = 100** Nivel de confianza mayor al 93% para los resultados de las pruebas del escenario GetEventHeader. Nivel de confianza mayor al 98% para los resultados de las pruebas del escenario GetEventSummaryData. Las pruebas siempre tuvieron un tiempo de respuesta menor a 1 segundo, mientras la utilización de procesador fue menor al 30%, la memoria disponible osciló entre 7GB y 8GB y el tamaño promedio de las solicitudes de I/O a disco fueron entre 10 y 20 KB.
GetEventHeader Confianza mayor al 93%.	
GetEvent SummaryData Confianza mayor al 98%.	
GetNotifications Confianza del 100%.	**Delta = 0.05 (5%), minMetric = 0.8, numRules = 100** En este escenario se involucran los llamados a 2 tipos de procesos: • GetNotificationHistoryCurrentUser. • GetNotificationDetails. El hardware y la aplicación se comportaron de igual manera que las pruebas anteriores. Las pruebas siempre tuvieron un tiempo de respuesta menor a 1 segundo, mientras la utilización de procesador fue menor al 30%, la memoria disponible osciló entre 7GB y 8GB y el tamaño promedio de las solicitudes de I/O a disco fueron entre 10 y 20 KB.
GetPendingServerEvents Confianza del 100%.	**Delta = 0.05 (5%), minMetric = 0.8, numRules = 100** Las pruebas siempre tuvieron un tiempo de respuesta menor a 1 segundo, mientras la utilización de procesador fue menor al 30%, la memoria disponible osciló entre 7GB y 8GB y el tamaño promedio de las solicitudes de I/O a disco fueron entre 10 y 20 KB.
ProcessTrace Results Confianza mayor al 97%.	

De esta manera, al segregar o dividir los escenarios de prueba para este caso en particular, se confirma que el comportamiento de la aplicación es muy bueno en un ambiente simulado de producción, con las características de hardware con las que se contará una vez que se ponga en marcha el sistema.

Se confirma y valida también que en este caso los resultados son similares a los obtenidos al analizar un único archivo con la consolidación de todas las pruebas aplicadas al sistema informático; por lo que el análisis de ellos no arroja resultados importantes o diferentes a los obtenidos anteriormente.

5.1.5 Verificación de los datos obtenidos por medio de tablas Pivote

En esta sección se lleva a cabo el análisis de los datos generados con base en los archivos de los logs del cliente y el servidor. Lo que se pretende es validar que las reglas de asociación obtenidas en las secciones anteriores corresponden adecuadamente a los resultados de los datos procesados. Para lograr esto se analizaron dos tipos de gráficos, el detalle de ellos a continuación.

5.1.5.1 Análisis del grafico "Número de ocurrencias por clasificación de tiempo de respuesta".

Este gráfico refleja el número de ocurrencias por clasificación de tiempo de respuesta, en donde se puede observar que para la mayoría de operaciones el tiempo de respuesta fue menor a 1 segundo. En contadas ocasiones el tiempo de respuesta fue mayor.

De acuerdo a la graficación de los resultados procesados de los archivos de logs y las reglas arrojadas luego de la aplicación del algoritmo Apriori, se puede validar y verificar que la información generada es acertada y correcta. Como se mencionó anteriormente, de la interpretación de las reglas de asociación generadas se puede deducir también que durante las pruebas de desempeño la aplicación en la mayoría de los casos obtuvo un tiempo de respuesta menor a 1 segundo, cumpliendo con los parámetros esperados para poder mover la aplicación a ambientes de producción.

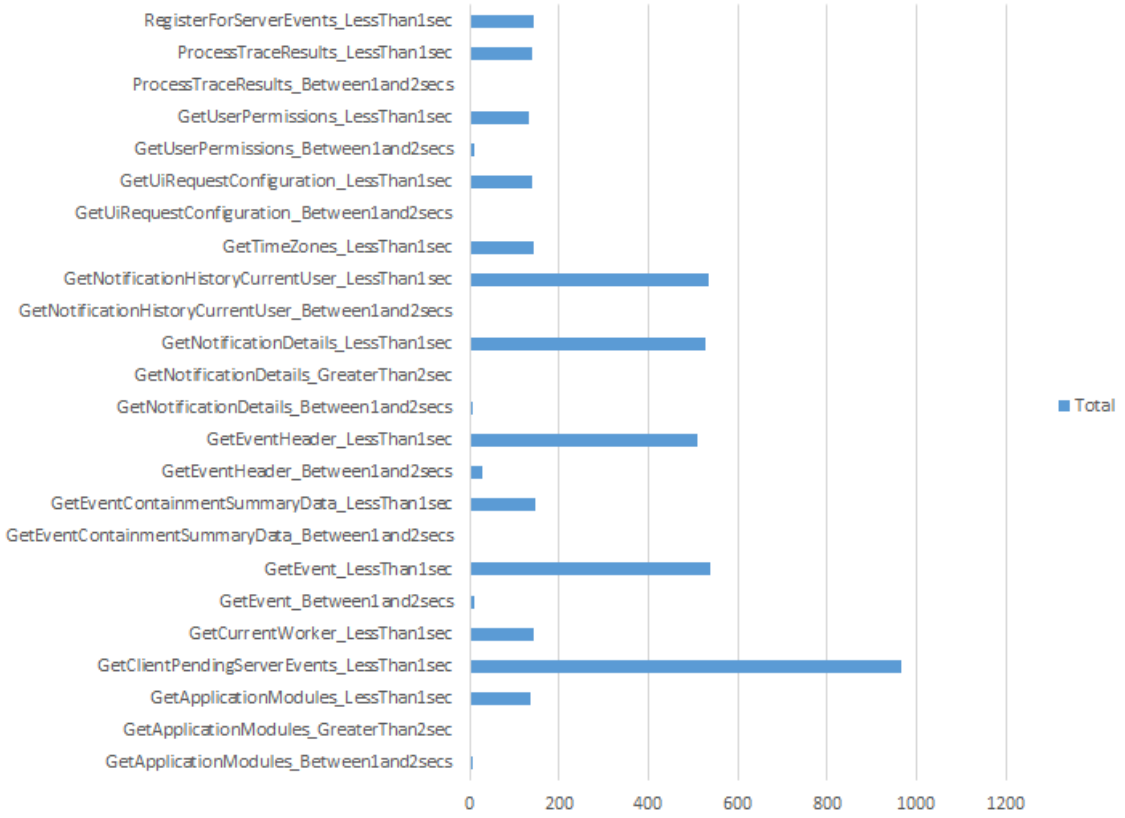

Figura 8. Número de ocurrencias por clasificación de tiempo de respuesta.

5.1.5.2 Análisis del gráfico "Medidas del hardware obtenido por escenario de prueba y operación".

En la figura 9 se muestran las medidas del hardware obtenidas por escenario de prueba y operación, se puede observar que el escenario de prueba que tuvo más ocurrencias fue la combinación de:

- Porcentaje de CPU utilizado menor a 30%.
- Memoria disponible entre 7GB y 8GB.
- Velocidad de la interface de Red menor a 20Kbs.
- Promedio de las solicitudes de I/O a disco fue entre 10KB y 20KB.

Los resultados del análisis de este gráfico concuerdan también con las reglas de asociación obtenidas. De igual forma, los resultados arrojados de los componentes de hardware a la

hora de ejecutar los escenarios de prueba cumplen con los requisitos para mover la aplicación a ambientes de producción.

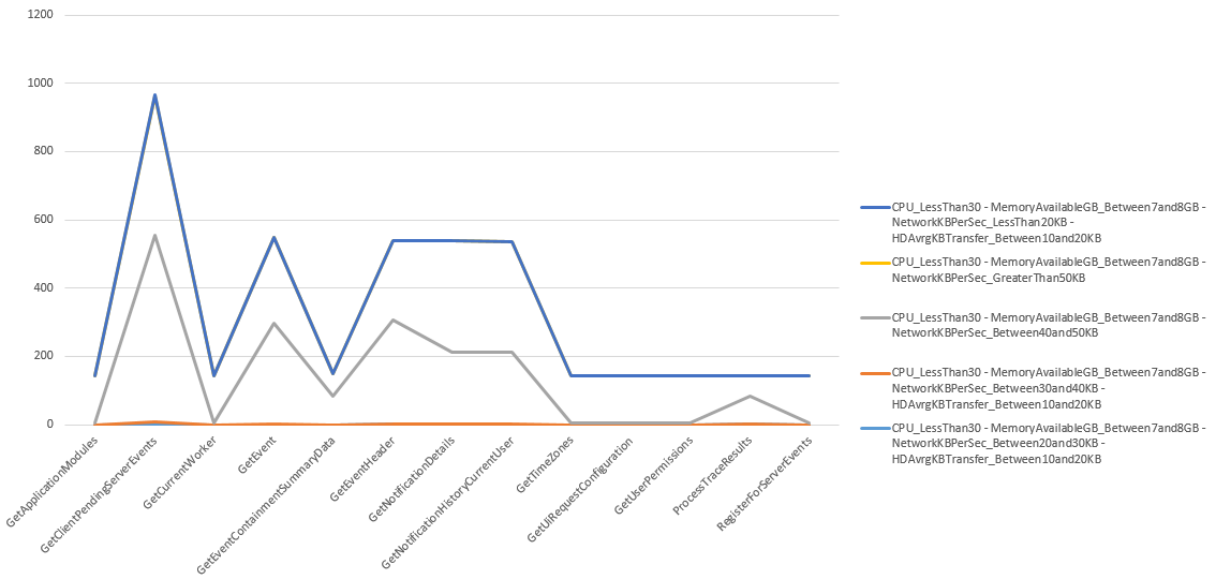

Figura 9. Medidas del hardware obtenido por escenario de prueba y operación.

De esta forma, se puede concluir que los resultados concuerdan con las reglas de asociación obtenidas y que el modelo propuesto para analizar los resultados de las pruebas funciona.

CAPÍTULO 6 Conclusiones y Recomendaciones

6.1 Conclusiones

De acuerdo con los resultados obtenidos luego de aplicar el modelo, podemos validar y comprobar que el sistema al que se le aplicaron las pruebas de **RAPS** se comportó estable y soporta de manera exitosa y sin problemas los escenarios ejecutados sobre los servidores con las características específicas de hardware de producción.

La característica de los datos recolectados de la aplicación probada no varía mucho durante el período de tiempo cuando se llevaron a cabo las pruebas, por lo que también se puede deducir que la confiabilidad de las reglas y resultados obtenidos son de alta confianza.

También se notó que si una determinada prueba durante el período de la aplicación de las misma se ejecuta pocas veces en comparación con los otros escenarios, el resultado de ella pasa a ser despreciable para la generación de las reglas; por lo que puede no ser tomado en cuenta por el algoritmo. Debido a esto, es muy importante definir de manera precisa la frecuencia de las pruebas aplicadas.

Otro punto importante a mencionar es que cuanto más cantidad de variables contenga la regla en “X” o en “Y”, hay más aporte al análisis de las pruebas. De lo contrario habrá que analizar mayor cantidad de reglas para tener una comprensión más completa y precisa de cómo sucedieron y se comportaron las pruebas.

Para el caso de la aplicación a la cual se le realizaron las pruebas en esta investigación, se evidencia que el resultado de la segregación de tipos de pruebas es muy similar al análisis previo del resultado de las pruebas, el cual fue almacenado en un solo archivo con la consolidación de todas las pruebas aplicadas. Aunque el resultado del análisis dependerá de las características de cada aplicación y su arquitectura; por lo que se podría decir que en otros escenarios sería válido segregar los resultados y hacer el análisis por tipo de prueba.

De esta manera, se obtendrían resultados más exactos, precisos y válidos sobre las pruebas aplicadas.

Por otro lado, otro tipo de resultados que podría arrojar el análisis de las reglas, sería tener resultados que indiquen que algo está sucediendo y está degradando el rendimiento de un sistema. Por ejemplo, una de las reglas generadas por el modelo podría indicar lo siguiente:

- ***server_percentage_processor_time = CPU_Between60and80***

  memory_available_gb = MemoryAvailableGB_Between1and2GB 4290 *==>*
  hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722
  conf:(0.90)

De acuerdo al hardware y los escenarios a los cuáles se están aplicando las pruebas, indicaría que algo anda mal en la aplicación. Dependiendo del hardware disponible y el escenario de prueba, se podrían tener conclusiones negativas, donde el rendimiento del sistema se ve comprometido.

Particularmente en este ejemplo, si nuestro servidor tiene entre sus especificaciones de memoria RAM 12 GB, pero en cierto momento la regla indica que solo hay entre 1 y 2 GB de memoria disponible; entonces, se podría interpretar que para ciertas pruebas la memoria no está siendo administrada de la mejor manera por la aplicación. También, el procesador está siendo utilizado entre un 60% y un 80%. Esto definitivamente levantaría una alerta para el equipo de desarrollo encargado de la aplicación.

El nivel de confianza es muy importante a la hora de interpretar la regla, ya que, en este caso en particular se está demostrando que la frecuencia con la que se da ese patrón fue con un 90 % de confianza. En este caso el nivel de confianza es muy alto, por lo que se puede determinar que la regla es muy confiable y veraz.

Hipotéticamente, también cabe destacar que las reglas podrían hacer alusión a un escenario de prueba específico; por ejemplo, la siguiente regla muestra que hay un problema con el tipo de prueba llamado "*GetNotificationDetails*".

- *responsetimeclassification=GetNotificationDetails_GreaterThan2secs*
  memory_available_gb=MemoryAvailableGB_Between2and3GB 529 ==>

operation=GetNotificationDetails
server_percentage_processor_time=CPU_LessThan30 529 conf:(1)

De esta regla, particularmente, se puede interpretar que para la prueba *"GetNotificationDetails"* la memoria RAM no está siendo manejada de la mejor manera por el sistema, ya que solo quedan entre 1 y 2 GB de memoria disponible para la ejecución de esa prueba. Además, el tiempo de respuesta esperado es de 1 o menos de 1 segundo, por lo que en esta regla no se estaría cumpliendo con el parámetro deseado para el tiempo de respuesta.

En casos como estos, los desarrolladores tendrán que trabajar de nuevo en la solución del problema encontrado, siendo esto un indicador de que la aplicación no puede ser puesta en producción; evitando que haya potencialmente interrupciones en los servicios brindados por la aplicación en cuestión.

6.1.1 Análisis Retrospectivo

Cuando se aborda el tema del desarrollo de software, indudablemente hay que referirse al ciclo de vida de desarrollo de sistemas informáticos y las etapas involucradas para obtener el producto final. Una de las etapas del ciclo, se refiere a las pruebas que se deben realizar antes de liberar un sistema a ambientes de producción. La fase de pruebas es uno de los enfoques de esta investigación. Las pruebas tienen el objetivo de garantizar que la aplicación se comportará y trabajará de manera adecuada una vez que sea puesta en marcha.

Por otro lado, en la actualidad nos enfrentamos al reto de procesar volúmenes de datos masivos y complejos; es necesario contar con herramientas, métodos o modelos que permitan el procesamiento eficiente de esos datos. Los resultados de las pruebas de rendimiento pueden llegar a producir grandes volúmenes de información, aunque es importante mencionar que esto dependerá exclusivamente de la complejidad de la aplicación, su arquitectura y la duración definida para la ejecución de las pruebas.

Este trabajo plantea un modelo a seguir donde por medio de la estandarización de procesos, herramientas de *Big Data* y la generación de reglas de asociación, permitan la comprensión del comportamiento de la aplicación durante la ejecución de las pruebas.

Indudablemente el modelo facilita el análisis y la comprensión del comportamiento de la aplicación por medio de las reglas de asociación generadas al final del modelo. La identificación de los patrones en los resultados indica que tan bien o mal se comportó el sistema cuando se llevaron a cabo las pruebas.

El ejercicio se llevó a cabo utilizando resultados de una aplicación crítica que sirvió para la comprobación y validación del modelo.

6.1.1.1 Limitaciones

El modelo planteado cumple con la hipótesis y los objetivos definidos en este investigación, sin embargo por motivos de alcance y tiempo para su desarrollo, el mismo fue probado con un solo sistema informático, por lo que la aplicación a otro tipo de sistemas con diferentes características, arquitectura y durante un mayor período de tiempo serían escenarios a tomar en cuenta para el modelo propuesto, además de generar mayor cantidad de información o datos que se puedan procesar.

Otro aspecto a importante a validar, es el modificar los escenarios o tipos de prueba definidos para que se ejecuten con mayor frecuencia y concurrencia, esto provocaría mayor consumo de recursos, por lo que se esperaría que el modelo al final produzca reglas de asociación que indiquen escenarios de riesgo que comprometen el despeño del sistema informático.

Por otro lado, el presente trabajo se enfocó a un solo tipo de algoritmo para la generación de las reglas de asociación, sin embargo, el estudio, comparación y mejoras a los diferentes algoritmos para la generación de las reglas, es un aspecto importante a tomar en consideración, ya que por la gran cantidad de información que podría ser analizada es necesario también evaluar este aspecto.

Finalmente, el modelo puede ser aún más genérico implementando herramientas para la generación automática del script hql, donde se tomen en cuenta todos los *performance counters* disponibles y basado en eso se puedan definir los diferentes tipos de clasificación a utilizar, al definir por ejemplo que valores corresponderían a Muy lento, Lento, Aceptable, Rápido o Muy Rápido por ejemplo. Donde, de igual forma se puedan configurar los escenarios de prueba y la clasificación correspondiente a cada tiempo de respuesta registrado.

6.2 Recomendaciones

Una vez finalizada la investigación y aplicado el modelo propuesto, se detectaron una serie de esfuerzos o estudios que podrían ser realizados como complemento a esta investigación, ente ellos se identificaron los siguientes:

- Creación de consola para simular las pruebas dependiendo de la aplicación. Esa aplicación debe generar los logs en el formato esperado y trabajado en esta investigación.
- Creación de Shell script de Linux para automatización del script de HQL creado para procesar y preparar los datos. De esa manera, se estaría automatizando la ejecución del script y la creación del *output* final esperado para la aplicación del algoritmo Apriori.
- Implementación para visualizar e interactuar con los resultados de una mejor manera. La visualización de los datos podría ser pensada para ser llevada a cabo en "tiempo real", de acuerdo a los datos que se irían obteniendo de las pruebas aplicadas. Una visualización amigable ayudaría y mejoraría indudablemente la interpretación de las reglas.
- Se puede pensar en persistir los resultados de los logs en bases de datos tradicionales o bases de datos No SQL.

- Al persistir los datos de las pruebas se podrían hacer análisis de resultados históricos de las pruebas y compararlas entre ellas para determinar el comportamiento de la aplicación en el tiempo de acuerdo a los cambios realizados.
- Aplicación de las mejoras al algoritmo Apriori. Esto ayudaría a que el proceso de la generación de las reglas pudiera ser más eficiente y eficaz.

Bibliografía

6 Top Data Analysis Tools for Big Data. (n.d.). Retrieved 7 22, 2013, from 6 Top Data Analysis Tools for Big Data: http://www.linuxlinks.com/article/20130406024357813/DataAnalysisTools.html

Analisis de asociaciones. (s.f.). Recuperado el 20 de Julio de 2013, de Analisis de asociaciones: http://ocw.univalle.edu.co/ocw/ingenieria-de-sistemas-telematica-y-afines/descubrimiento-de-conocimiento-en-bases-de-datos/material-1/Asociacion.pdf

Anand H.S., V. S. (2013). Applying Correlation Threshold on Apriori Algorithm. *IEEE International Conference on Emerging Trends in Computing, Communication and Nanotechnology (ICECCN 2013)*, 4.

Ansel Yoan Rodríguez González, e. a. (31 de Marzo de 2009). *Mineria de reglas de asociacion sobre datos mezclados*. Recuperado el 30 de Julio de 2013, de Mineria de reglas de asociacion sobre datos mezclados: http://ccc.inaoep.mx/portalfiles/file/CCC-09-001.pdf

Apriori algorithm for Data Mining - made simple. (n.d.). Retrieved Julio 28, 2013, from Apriori: Este algoritmo se usa en minería de datos para encontrar Reglas de asociación en un conjunto de datos. Este algoritmo se basa en el conocimiento previo o “a priori” de los conjuntos frecuentes, esto sirve para reducir el espacio de búsqueda y aum

Atlassian. (28 de Setiembre de 2015). Obtenido de https://cwiki.apache.org/confluence/display/Hive/HiveServer2+Clients

Barber, S. (2006). *Performance Testing Challenges*. Recuperado el 15 de 6 de 2013, de Performance Testing Challenges: http://www.perftestplus.com/resources/perf_challenges_ppt.pdf

Barber, S. (n.d.). *Performance Testing Challenges*. Retrieved Julio 10, 2013, from Performance Testing Challenges: http://www.perftestplus.com/resources/perf_challenges_ppt.pdf

Cassandra, P. (26 de August de 2015). *What is Apache Cassandra?* Obtenido de http://www.planetcassandra.org/what-is-apache-cassandra/

Datamation. (24 de August de 2015). *50 Top Open Source Tools for Big Data*. Obtenido de http://www.datamation.com/data-center/50-top-open-source-tools-for-big-data-1.html

Extended WEKA including Ensembles of Hierarchically Nested Dichotomies. (19 de Setiembre de 2015). Obtenido de http://www.dbs.ifi.lmu.de/~zimek/diplomathesis/implementations/EHNDs/doc/weka/associations/Apriori.html

Geroba. (26 de August de 2015). *Cassandra Introduction & Features*. Obtenido de SlideShare: http://es.slideshare.net/planetcassandra/cassandra-introduction-features-30103666

Hadoop, A. (n.d.). *Apache Hadoop*. Retrieved 8 5, 2013, from Apache Hadoop: http://hadoop.apache.org/

Hahsler, M. (19 de Setiembre de 2015). *A Probabilistic Comparison of Commonly Used Interest Measures for Association Rules.* Obtenido de http://michael.hahsler.net/research/association_rules/measures.html#leverage

Handler, B. (26 de August de 2015). *What is Hadoop?* Obtenido de http://bigdatahandler.com/hadoop-hdfs/what-is-apache-hadoop/

Handler, B. D. (26 de August de 2015). *What is Apache Hadoop?* Obtenido de http://bigdatahandler.com/hadoop-hdfs/what-is-apache-hadoop/

Introduction to Datamining. (n.d.). Retrieved Agosto 5, 2013, from Introduction to Datamining: http://www-users.cs.umn.edu/~kumar/dmbook/index.php

Juan Camilo Giraldo Mejía, e. a. (2012). Aplicación de la técnica de asociación de la minería de datos en un caso de investigación. *Revista Nacional de Investigacion-Memorias, Voplumen 10, Numero 18/julio-diciembre 2012*.

King Chun Foo, e. a. (2010). Mining Performance Regression Testing Repositories for Automated Performance Analysis. *10th International Conference on Quality Software*, 10.

Lei Ji, e. a. (2006). A New Improvement on Apriori Algorithm. *IEEE*, 5.

Lucas, J. P. (2 de Noviembre de 2015). *Métodos de clasificación basados en asociación aplicados a sistemas de recomendación.* Obtenido de http://gredos.usal.es/xmlui/bitstream/handle/10366/83342/DIA_PinhoLucasJ_M%C3%A9todosdeclasificaci%C3%B3n.pdf?sequence=1

MetaScale. (2013). *Why Hadoop*. Retrieved Agosto 20, 2013, from Why Hadoop: http://www.metascale.com/why-metascale/why-hadoop

Miao Jiang, e. a. (2009). System Monitoring with Metric-Correlation Models: Problems and Solutions. *ACM 978-1-60558-564-2/09/06*, 10.

Microsoft. (12 de Junio de 2013). *Performance Counters*. Recuperado el 15 de Agosto de 2013, de http://msdn.microsoft.com/en-us/library/windows/desktop/aa373083(v=vs.85).aspx

Microsoft. (19 de Setiembre de 2015). *Performance data for Bytes Total/sec counter*. Obtenido de https://technet.microsoft.com/en-us/library/aa996576(v=exchg.80).aspx

Microsoft. (19 de Setiembre de 2015). *Windows Performance Monitor Disk Counters Explained*. Obtenido de http://blogs.technet.com/b/askcore/archive/2012/03/16/windows-performance-monitor-disk-counters-explained.aspx

mongoDB. (23 de August de 2015). *mongoDB*. Obtenido de Introduction to MongoDB: https://www.mongodb.org/about/introduction/

Patrick Reynolds, e. a. (2006). WAP5: Blackbox Performance Debugging for WideArea Systems. . *15th InternationalWorld WideWeb Conference*, 10.

Raudel Hernandez Leaon, e. a. (Agosto de 2010). *Descubrimiento de conjuntos frecuentes de itemes en datos estaticos y dinamicos*. Recuperado el 11 de Agosto de 2013, de http://www.cenatav.co.cu/doc/RTecnicos/RT%20SerieGris_006web.pdf

Sistemas de soporte a la toma de decisiones. (15 de Diciembre de 2010). Recuperado el 27 de Julio de 2013, de Sistemas de soporte a la toma de decisiones: http://dataminingfime.blogspot.com/2010/12/reglas-se-asociacion.html

Stephen Kaisler, e. a. (2013). Big Data: Issues and Challenges Moving Forward. *46th Hawaii International Conference on System Sciences*.

Tools, A. (19 de Setiembre de 2015). *Windows Performance Counters Explained*. Obtenido de http://www.appadmintools.com/documents/winperf_counters_cpu.html

Tools, A. (19 de Setiembre de 2015). *Windows Performance Counters Explained*. Obtenido de http://www.appadmintools.com/documents/winperf_counters_memory.html

Vaibhav Sharma, M. S. (2010). A Probabilistic Approach to Apriori Algorithm. *2010 IEEE International Conference on Granular Computing*, 7.

Waikato., U. o. (15 de Setiembre de 2015). *Weka*. Obtenido de http://www.cs.waikato.ac.nz/ml/weka/

Wang, X. (2010). Study of Data Ming based on Apriori Algorithm. *2nd International Conference on Software Technology and Engineering(ICSTE)*, 4.

Weixiao LIU, e. a. (n.d.). An Improved Apriori Algorithm.

Wikipedia. (18 de Octubre de 2016). *Cloudera*. Obtenido de https://es.wikipedia.org/wiki/Cloudera

Wikipedia. (s.f.). *Algoritmo apriori*. Recuperado el 18 de Julio de 2013, de Algoritmo apriori: http://es.wikipedia.org/wiki/Algoritmo_apriori

Wikipedia. (s.f.). *Category: Software testing tools*. Recuperado el 22 de Agosto de 2013, de Category: Software testing tools: http://en.wikipedia.org/wiki/Category:Software_testing_tools

Wikipedia. (n.d.). *Cross Industry Standard Process for Data Mining*. Retrieved 7 11, 2013, from Cross Industry Standard Process for Data Mining: http://en.wikipedia.org/wiki/CRISP_DM

Wikipedia. (n.d.). *Reglas de Asociacion*. Retrieved 7 2013, 15, from Reglas de Asociacion: http://es.wikipedia.org/wiki/Reglas_de_asociaci%C3%B3n

Wikipedia. (n.d.). *Software performance testing*. Retrieved Agosto 1, 2013, from Software performance testing: http://en.wikipedia.org/wiki/Software_performance_testing

Yiwu Xie, e. a. (2008). The Optimization and Improvement of the Apriori Algorithm. *International Symposium on Intelligent Information Technology Application Workshops*, 3.

Anexos

Anexo A. Formateo y limpia de los datos por medio de un HQL Script.

```
-- ********************************
-- * BORRADO DE LAS TABLAS.   *
-- ********************************
-- Borra las tablas relacionadas al log del cliente.
drop table ClientLogClassified;
drop table ClientLogGroupedbyDatetime;
drop table ClientLogSecondsRemoved;
drop table ClientLog;  -- This is not necessary to delete to start the process.
-- Borra las tablas relacionadas al log del servidor.
drop table ServerLogClassified;
drop table ServerLogMeasurementConverted;
drop table serverloggroupedbydatetime;
drop table ServerLogSecondsRemoved;
drop table serverlog; -- This is not necessary to delete to start the process.;
drop table finallog;
-- ***********************************
-- *  PODA DEL LOG DEL CLIENTE.  *
-- ***********************************
-- Creación de la tabla ClientLog
CREATE TABLE IF NOT EXISTS ClientLog(
Timestamp string,
RequestId string,
Operation string,
Status  string,
ResponseTime decimal(10,2),
ExceptionMessage string)
COMMENT 'Client RAPS Log table'
```

```
ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

STORED AS TEXTFILE;

-- Remueve los segundos del timestamp.

-- Response time es convertido de milisegundos a segundos.

-- Resultado guardado en una nueva tabla llamada ClientLogSecondsRemoved.

 Create Table ClientLogSecondsRemoved AS

select substring(timestamp,1,length(timestamp)-3) as timestamp, operation, responsetime/1000 as
responsetimesecs

from clientlog;

-- Creación de una tabla nueva con los datos agrupados por timestamp.

-- Para cada grupo se calcula el promedio de responsetime.

 Create table ClientLogGroupedbyDatetime As

Select timestamp, operation,  round((sum(responsetimesecs)/count(*)),2) as AvrgRespTimeSecs

from ClientLogSecondsRemoved

group by timestamp, operation;

-- Creación de nueva tabla ClientLogClassified.

-- Esta tabla contiene la clasificación de cada registro de acuerdo al responsetime

Create table ClientLogClassified As

Select timestamp, operation, avrgresptimesecs,

        case

                when AvrgRespTimeSecs<1 then CONCAT(operation, '_LessThan1sec')

                when AvrgRespTimeSecs>1 and AvrgRespTimeSecs<2 then CONCAT(operation,
'_Between1and2secs')

                when AvrgRespTimeSecs<2 then CONCAT(operation, '_GreaterThan2sec')

     end as classification

from   ClientLogGroupedbyDatetime;

-- **********************************

-- *  PODA DEL LOG DEL SERVIDOR.  *

-- **********************************
```

-- Creación de la tabla ServerLog.

```
CREATE TABLE IF NOT EXISTS ServerLog(
Timestamp string,
Server_Percentage_Processor_Time decimal(10,5),
Memory_Available_Bytes decimal(20,5),
Network_Bytes_TotalPersec decimal(20,5),
HD_Avg_DiskBytesTransfer decimal(20,5),
HD_Avg_DiskQueueLength decimal(20,5))
COMMENT 'Server RAPS Logs'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

-- Remueve segundos del timestamp.

-- El resultado se almacena en una nueva tabla llamada ServerLogSecondsRemoved.

```
Create Table ServerLogSecondsRemoved AS
select substring(timestamp,1,length(timestamp)-3) as timestamp,
    Server_Percentage_Processor_Time,
    Memory_Available_Bytes,
    Network_Bytes_TotalPersec,
    HD_Avg_DiskBytesTransfer,
    HD_Avg_DiskQueueLength
from   serverlog
where  timestamp<>'timestamp';
```

-- Se agrupan los registros por timestamp y se calcula el promedio para cada medida (CPU, Memoria, Red, Disco Duro).

-- El resultado se almacena en una nueva tabla llamada serverloggroupedbydatetime.

```
Create table serverloggroupedbydatetime As
select   timestamp,
      round(sum(Server_Percentage_Processor_Time)/count(*), 2) As Server_Percentage_Processor_Time,
      round(sum(Memory_Available_Bytes)/count(*), 2) As Memory_Available_Bytes,
```

```
        round(sum(Network_Bytes_TotalPersec)/count(*), 2) As Network_Bytes_TotalPersec,
        round(sum(HD_Avg_DiskBytesTransfer)/count(*), 2) As HD_Avg_DiskBytesTransfer,
        round(sum(HD_Avg_DiskQueueLength)/count(*), 2) As HD_Avg_DiskQueueLength
from     ServerLogSecondsRemoved
group by timestamp;
```

-- Se convierten las unidades de cada medida.

-- El resultado se guarda en una nueva tabla llamada ServerLogMeasurementConverted.

```
Create Table ServerLogMeasurementConverted as
select  timestamp,
        server_percentage_processor_time,
        (((memory_available_bytes/1024)/1024)/1024) as memory_available_GB,
        (network_bytes_totalpersec/1024) as network_KB_totalpersec,
        (hd_avg_diskbytestransfer/1024) as hd_avg_disk_KB_transfer
from    serverloggroupedbydatetime;
```

-- Se clasifica cada registro de acuerdo a las medidas obtenidas.

-- El resultado se almacena en una nueva tabla llamada ServerLogClassified.

```
Create Table ServerLogClassified As
Select     timestamp,
                    -- CPU Classificaation.
                    case
                                when server_percentage_processor_time<=30 then 'CPU_LessThan30'
                                when (server_percentage_processor_time>30 and
server_percentage_processor_time<=60) then 'CPU_Between30and60'
                                when (server_percentage_processor_time>60 and
server_percentage_processor_time<=80) then 'CPU_Between60and80'
                                when server_percentage_processor_time>80 then 'CPU_GraterThan80'
                    end as server_percentage_processor_time,
                    -- Memory Classification.
                    case
                                when memory_available_gb<=1 then 'MemoryAvailableGB_LessThan1'
                                when (memory_available_gb>1 and memory_available_gb<=2) then
'MemoryAvailableGB_Between1and2GB'
```

```
                when (memory_available_gb>2 and memory_available_gb<=3) then
'MemoryAvailableGB_Between2and3GB'

                when (memory_available_gb>3 and memory_available_gb<=4) then
'MemoryAvailableGB_Between3and4GB'

                when (memory_available_gb>4 and memory_available_gb<=5) then
'MemoryAvailableGB_Between4and5GB'

                when (memory_available_gb>5 and memory_available_gb<=6) then
'MemoryAvailableGB_Between5and6GB'

                when (memory_available_gb>6 and memory_available_gb<=7) then
'MemoryAvailableGB_Between6and7GB'

                when (memory_available_gb>7 and memory_available_gb<=8) then
'MemoryAvailableGB_Between7and8GB'

                when (memory_available_gb>8 and memory_available_gb<=9) then
'MemoryAvailableGB_Between8and9GB'

                when (memory_available_gb>9 and memory_available_gb<=10) then
'MemoryAvailableGB_Between9and10GB'

                when memory_available_gb>10 then 'MemoryAvailableGB_GraterThan10'


        end as memory_available_GB,

    -- Network.

        case

                when network_KB_totalpersec<=20 then 'NetworkKBPerSec_LessThan20KB'

                when (network_KB_totalpersec>20 and network_KB_totalpersec <=30) then
'NetworkKBPerSec_Between20and30KB'

                when (network_KB_totalpersec>30 and network_KB_totalpersec <=40) then
'NetworkKBPerSec_Between30and40KB'

                when (network_KB_totalpersec>40 and network_KB_totalpersec <=50) then
'NetworkKBPerSec_Between40and50KB'

                when network_KB_totalpersec>50 then 'NetworkKBPerSec_GreaterThan50KB'

        end as network_KB_totalpersec,

        -- Hard Drive

    case

                when hd_avg_disk_KB_transfer<=10 then 'HDAvrgKBTransfer_LessThan10KB'

                when (hd_avg_disk_KB_transfer>10 and hd_avg_disk_KB_transfer<=20) then
'HDAvrgKBTransfer_Between10and20KB'

                when (hd_avg_disk_KB_transfer>20 and hd_avg_disk_KB_transfer<=30) then
'HDAvrgKBTransfer_Between20and30KB'
```

```
                when (hd_avg_disk_KB_transfer>30 and hd_avg_disk_KB_transfer<=40) then
'HDAvrgKBTransfer_Between30and40KB'

                when (hd_avg_disk_KB_transfer>40 and hd_avg_disk_KB_transfer<=50) then
'HDAvrgKBTransfer_Between40and50KB'

                when (hd_avg_disk_KB_transfer>50 and hd_avg_disk_KB_transfer<=60) then
'HDAvrgKBTransfer_Between50and60KB'

                when (hd_avg_disk_KB_transfer>60 and hd_avg_disk_KB_transfer<=70) then
'HDAvrgKBTransfer_Between60and70KB'

                when (hd_avg_disk_KB_transfer>70 and hd_avg_disk_KB_transfer<=80) then
'HDAvrgKBTransfer_Between70and80KB'

                when (hd_avg_disk_KB_transfer>80 and hd_avg_disk_KB_transfer<=90) then
'HDAvrgKBTransfer_Between80and90KB'

                when (hd_avg_disk_KB_transfer>90 and hd_avg_disk_KB_transfer<=100) then
'HDAvrgKBTransfer_Between90and100KB'

                when hd_avg_disk_KB_transfer>100 then
'HDAvrgKBTransfer_GreaterThan100KB'

        end as hd_avg_disk_KB_transfer

from    ServerLogMeasurementConverted;
```

-- Se crea un join entre las dos tablas finales con el fin de obtener una sola tabla.

-- De esta manera se generará un solo archivo basado en la tabla final.

-- Este archivo será el input para el algoritmo APRIORI.

-- El resultado final es almacenado en una tabla llamada FinalLog.

```
Create Table FinalLog As

SELECT          ClientLogClassified.operation,

                ClientLogClassified.classification as ResponseTimeClassification,

                ServerLogClassified.server_percentage_processor_time,

                ServerLogClassified.memory_available_GB,

                ServerLogClassified.network_KB_totalpersec,

                ServerLogClassified.hd_avg_disk_KB_transfer

FROM    ClientLogClassified

                JOIN ServerLogClassified on
ClientLogClassified.timestamp=ServerLogClassified.timestamp;
```

Anexo B. Reglas encontradas para el escenario "Delta = 0.05 (5%), minMetric = 0.9, numRules = 10".

1. memory_available_gb=MemoryAvailableGB_Between7and8GB 4290 ==>
server_percentage_processor_time=CPU_LessThan30 4290 conf:(1)

2. server_percentage_processor_time=CPU_LessThan30 4290 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 4290 conf:(1)

3. hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
server_percentage_processor_time=CPU_LessThan30 3722 conf:(1)

4. hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 3722 conf:(1)

5. memory_available_gb=MemoryAvailableGB_Between7and8GB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
server_percentage_processor_time=CPU_LessThan30 3722 conf:(1)

6. server_percentage_processor_time=CPU_LessThan30
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 3722 conf:(1)

7. hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 3722 conf:(1)

8. network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB 2502 ==>
server_percentage_processor_time=CPU_LessThan30 2502 conf:(1)

9. network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB 2502 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 2502 conf:(1)

10. memory_available_gb=MemoryAvailableGB_Between7and8GB
network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB 2502 ==>
server_percentage_processor_time=CPU_LessThan30 2502 conf:(1)

Anexo C. Reglas encontradas para el escenario "Delta = 0.05 (5%), minMetric = 0.7, numRules = 10".

1. memory_available_gb=MemoryAvailableGB_Between7and8GB 4290 ==>
server_percentage_processor_time=CPU_LessThan30 4290 conf:(1)

2. server_percentage_processor_time=CPU_LessThan30 4290 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 4290 conf:(1)

3. hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
server_percentage_processor_time=CPU_LessThan30 3722 conf:(1)

4. hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 3722 conf:(1)

5. memory_available_gb=MemoryAvailableGB_Between7and8GB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
server_percentage_processor_time=CPU_LessThan30 3722 conf:(1)

6. server_percentage_processor_time=CPU_LessThan30
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 3722 conf:(1)

7. hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 3722 conf:(1)

8. server_percentage_processor_time=CPU_LessThan30 4290 ==>
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 conf:(0.87)

9. memory_available_gb=MemoryAvailableGB_Between7and8GB 4290 ==>
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 conf:(0.87)

10. server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 4290 ==>
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 conf:(0.87)

Anexo D. Reglas encontradas para el escenario "Delta = 0.05 (5%), minMetric = entre 0.2 y 0.6, numRules = 10".

1. memory_available_gb=MemoryAvailableGB_Between7and8GB 4290 ==>
server_percentage_processor_time=CPU_LessThan30 4290 conf:(1)

2. server_percentage_processor_time=CPU_LessThan30 4290 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 4290 conf:(1)

3. hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
server_percentage_processor_time=CPU_LessThan30 3722 conf:(1)

4. hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 3722 conf:(1)

5. memory_available_gb=MemoryAvailableGB_Between7and8GB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
server_percentage_processor_time=CPU_LessThan30 3722 conf:(1)

6. server_percentage_processor_time=CPU_LessThan30
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 3722 conf:(1)

7. hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 3722 conf:(1)

8. server_percentage_processor_time=CPU_LessThan30 4290 ==>
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 conf:(0.87)

9. memory_available_gb=MemoryAvailableGB_Between7and8GB 4290 ==>
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 conf:(0.87)

10. server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 4290 ==>
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 conf:(0.87)

Anexo E. Reglas encontradas para el escenario "Delta = 0.05 (5%), minMetric = 0.9, numRules = 50".

1. memory_available_gb=MemoryAvailableGB_Between7and8GB 4290 ==>
server_percentage_processor_time=CPU_LessThan30 4290 conf:(1)

2. server_percentage_processor_time=CPU_LessThan30 4290 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 4290 conf:(1)

3. hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
server_percentage_processor_time=CPU_LessThan30 3722 conf:(1)

4. hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 3722 conf:(1)

5. memory_available_gb=MemoryAvailableGB_Between7and8GB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
server_percentage_processor_time=CPU_LessThan30 3722 conf:(1)

6. server_percentage_processor_time=CPU_LessThan30
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 3722 conf:(1)

7. hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 3722 conf:(1)

8. network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB 2502 ==>
server_percentage_processor_time=CPU_LessThan30 2502 conf:(1)

9. network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB 2502 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 2502 conf:(1)

10. memory_available_gb=MemoryAvailableGB_Between7and8GB
network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB 2502 ==>
server_percentage_processor_time=CPU_LessThan30 2502 conf:(1)

11. server_percentage_processor_time=CPU_LessThan30
network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB 2502 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 2502 conf:(1)

12. network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB 2502 ==>
server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 2502 conf:(1)

13. network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 2148 ==> server_percentage_processor_time=CPU_LessThan30 2148 conf:(1)

14. network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 2148 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 2148 conf:(1)

15. memory_available_gb=MemoryAvailableGB_Between7and8GB network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 2148 ==> server_percentage_processor_time=CPU_LessThan30 2148 conf:(1)

16. server_percentage_processor_time=CPU_LessThan30 network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 2148 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 2148 conf:(1)

17. network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 2148 ==> server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 2148 conf:(1)

18. network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 1770 ==> server_percentage_processor_time=CPU_LessThan30 1770 conf:(1)

19. network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 1770 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 1770 conf:(1)

20. memory_available_gb=MemoryAvailableGB_Between7and8GB network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 1770 ==> server_percentage_processor_time=CPU_LessThan30 1770 conf:(1)

21. server_percentage_processor_time=CPU_LessThan30 network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 1770 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 1770 conf:(1)

22. network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 1770 ==> server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 1770 conf:(1)

23. network_KB_totalpersec=NetworkKBPerSec_Between40and50KB hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 1556 ==> server_percentage_processor_time=CPU_LessThan30 1556 conf:(1)

24. network_KB_totalpersec=NetworkKBPerSec_Between40and50KB hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 1556 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 1556 conf:(1)

25. memory_available_gb=MemoryAvailableGB_Between7and8GB network_KB_totalpersec=NetworkKBPerSec_Between40and50KB hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 1556 ==> server_percentage_processor_time=CPU_LessThan30 1556 conf:(1)

26. server_percentage_processor_time=CPU_LessThan30 network_KB_totalpersec=NetworkKBPerSec_Between40and50KB hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 1556 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 1556 conf:(1)

27. network_KB_totalpersec=NetworkKBPerSec_Between40and50KB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 1556 ==>
server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 1556 conf:(1)

28. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 ==>
operation=GetClientPendingServerEvents 967 conf:(1)

29. operation=GetClientPendingServerEvents 967 ==>
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 conf:(1)

30. operation=GetClientPendingServerEvents 967 ==> server_percentage_processor_time=CPU_LessThan30
967 conf:(1)

31. operation=GetClientPendingServerEvents 967 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

32. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 ==>
server_percentage_processor_time=CPU_LessThan30 967 conf:(1)

33. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

34. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
server_percentage_processor_time=CPU_LessThan30 967 ==> operation=GetClientPendingServerEvents
967 conf:(1)

35. operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30 967 ==>
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 conf:(1)

36. operation=GetClientPendingServerEvents
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 ==>
server_percentage_processor_time=CPU_LessThan30 967 conf:(1)

37. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 ==>
operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30 967
conf:(1)

38. operation=GetClientPendingServerEvents 967 ==>
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
server_percentage_processor_time=CPU_LessThan30 967 conf:(1)

39. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
memory_available_gb=MemoryAvailableGB_Between7and8GB 967 ==>
operation=GetClientPendingServerEvents 967 conf:(1)

40. operation=GetClientPendingServerEvents
memory_available_gb=MemoryAvailableGB_Between7and8GB 967 ==>
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 conf:(1)

41. operation=GetClientPendingServerEvents
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

42. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 ==>
operation=GetClientPendingServerEvents memory_available_gb=MemoryAvailableGB_Between7and8GB
967 conf:(1)

43. operation=GetClientPendingServerEvents 967 ==>
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

44. operation=GetClientPendingServerEvents
memory_available_gb=MemoryAvailableGB_Between7and8GB 967 ==>
server_percentage_processor_time=CPU_LessThan30 967 conf:(1)

45. operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30 967 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

46. operation=GetClientPendingServerEvents 967 ==> server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

47. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
memory_available_gb=MemoryAvailableGB_Between7and8GB 967 ==>
server_percentage_processor_time=CPU_LessThan30 967 conf:(1)

48. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
server_percentage_processor_time=CPU_LessThan30 967 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

49. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 ==>
server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

50. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 967 ==>
operation=GetClientPendingServerEvents 967 conf:(1)

Anexo F. Reglas encontradas para el escenario "Delta = 0.05 (5%), minMetric = 0.9, numRules = 200".

1. memory_available_gb=MemoryAvailableGB_Between7and8GB 4290 ==>
server_percentage_processor_time=CPU_LessThan30 4290 conf:(1)

2. server_percentage_processor_time=CPU_LessThan30 4290 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 4290 conf:(1)

3. hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
server_percentage_processor_time=CPU_LessThan30 3722 conf:(1)

4. hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 3722 conf:(1)

5. memory_available_gb=MemoryAvailableGB_Between7and8GB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
server_percentage_processor_time=CPU_LessThan30 3722 conf:(1)

6. server_percentage_processor_time=CPU_LessThan30
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 3722 conf:(1)

7. hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 3722 ==>
server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 3722 conf:(1)

8. network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB 2502 ==>
server_percentage_processor_time=CPU_LessThan30 2502 conf:(1)

9. network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB 2502 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 2502 conf:(1)

10. memory_available_gb=MemoryAvailableGB_Between7and8GB
network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB 2502 ==>
server_percentage_processor_time=CPU_LessThan30 2502 conf:(1)

11. server_percentage_processor_time=CPU_LessThan30
network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB 2502 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 2502 conf:(1)

12. network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB 2502 ==>
server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 2502 conf:(1)

13. network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 2148 ==>
server_percentage_processor_time=CPU_LessThan30 2148 conf:(1)

14. network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 2148 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 2148 conf:(1)

15. memory_available_gb=MemoryAvailableGB_Between7and8GB
network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 2148 ==>
server_percentage_processor_time=CPU_LessThan30 2148 conf:(1)

16. server_percentage_processor_time=CPU_LessThan30
network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 2148 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 2148 conf:(1)

17. network_KB_totalpersec=NetworkKBPerSec_GreaterThan50KB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 2148 ==>
server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 2148 conf:(1)

18. network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 1770 ==>
server_percentage_processor_time=CPU_LessThan30 1770 conf:(1)

19. network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 1770 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 1770 conf:(1)

20. memory_available_gb=MemoryAvailableGB_Between7and8GB
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 1770 ==>
server_percentage_processor_time=CPU_LessThan30 1770 conf:(1)

21. server_percentage_processor_time=CPU_LessThan30
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 1770 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 1770 conf:(1)

22. network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 1770 ==>
server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 1770 conf:(1)

23. network_KB_totalpersec=NetworkKBPerSec_Between40and50KB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 1556 ==>
server_percentage_processor_time=CPU_LessThan30 1556 conf:(1)

24. network_KB_totalpersec=NetworkKBPerSec_Between40and50KB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 1556 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 1556 conf:(1)

25. memory_available_gb=MemoryAvailableGB_Between7and8GB
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 1556 ==>
server_percentage_processor_time=CPU_LessThan30 1556 conf:(1)

26. server_percentage_processor_time=CPU_LessThan30
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 1556 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 1556 conf:(1)

27. network_KB_totalpersec=NetworkKBPerSec_Between40and50KB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 1556 ==>
server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 1556 conf:(1)

28. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 ==>
operation=GetClientPendingServerEvents 967 conf:(1)

29. operation=GetClientPendingServerEvents 967 ==>
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 conf:(1)

30. operation=GetClientPendingServerEvents 967 ==>
server_percentage_processor_time=CPU_LessThan30 967 conf:(1)

31. operation=GetClientPendingServerEvents 967 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

32. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 ==>
server_percentage_processor_time=CPU_LessThan30 967 conf:(1)

33. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

34. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
server_percentage_processor_time=CPU_LessThan30 967 ==> operation=GetClientPendingServerEvents
967 conf:(1)

35. operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30 967
==> responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 conf:(1)

36. operation=GetClientPendingServerEvents
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 ==>
server_percentage_processor_time=CPU_LessThan30 967 conf:(1)

37. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 ==> operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30 967 conf:(1)

38. operation=GetClientPendingServerEvents 967 ==> responsetimeclassification=GetClientPendingServerEvents_LessThan1sec server_percentage_processor_time=CPU_LessThan30 967 conf:(1)

39. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec memory_available_gb=MemoryAvailableGB_Between7and8GB 967 ==> operation=GetClientPendingServerEvents 967 conf:(1)

40. operation=GetClientPendingServerEvents memory_available_gb=MemoryAvailableGB_Between7and8GB 967 ==> responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 conf:(1)

41. operation=GetClientPendingServerEvents responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

42. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 ==> operation=GetClientPendingServerEvents memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

43. operation=GetClientPendingServerEvents 967 ==> responsetimeclassification=GetClientPendingServerEvents_LessThan1sec memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

44. operation=GetClientPendingServerEvents memory_available_gb=MemoryAvailableGB_Between7and8GB 967 ==> server_percentage_processor_time=CPU_LessThan30 967 conf:(1)

45. operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30 967 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

46. operation=GetClientPendingServerEvents 967 ==> server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

47. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec memory_available_gb=MemoryAvailableGB_Between7and8GB 967 ==> server_percentage_processor_time=CPU_LessThan30 967 conf:(1)

48. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec server_percentage_processor_time=CPU_LessThan30 967 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

49. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 ==> server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

50. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 967 ==> operation=GetClientPendingServerEvents 967 conf:(1)

51. operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 967 ==> responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 conf:(1)

52. operation=GetClientPendingServerEvents responsetimeclassification=GetClientPendingServerEvents_LessThan1sec memory_available_gb=MemoryAvailableGB_Between7and8GB 967 ==> server_percentage_processor_time=CPU_LessThan30 967 conf:(1)

53. operation=GetClientPendingServerEvents responsetimeclassification=GetClientPendingServerEvents_LessThan1sec server_percentage_processor_time=CPU_LessThan30 967 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

54. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec memory_available_gb=MemoryAvailableGB_Between7and8GB 967 ==> operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30 967 conf:(1)

55. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec server_percentage_processor_time=CPU_LessThan30 967 ==> operation=GetClientPendingServerEvents memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

56. operation=GetClientPendingServerEvents memory_available_gb=MemoryAvailableGB_Between7and8GB 967 ==> responsetimeclassification=GetClientPendingServerEvents_LessThan1sec server_percentage_processor_time=CPU_LessThan30 967 conf:(1)

57. operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30 967 ==> responsetimeclassification=GetClientPendingServerEvents_LessThan1sec memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

58. operation=GetClientPendingServerEvents responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 ==> server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

59. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 967 ==> operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

60. operation=GetClientPendingServerEvents 967 ==> responsetimeclassification=GetClientPendingServerEvents_LessThan1sec server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 967 conf:(1)

61. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==> operation=GetClientPendingServerEvents 831 conf:(1)

62. operation=GetClientPendingServerEvents hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==> responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 831 conf:(1)

63. operation=GetClientPendingServerEvents hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==> server_percentage_processor_time=CPU_LessThan30 831 conf:(1)

64. operation=GetClientPendingServerEvents
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 831 conf:(1)

65. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
server_percentage_processor_time=CPU_LessThan30 831 conf:(1)

66. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 831 conf:(1)

67. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
server_percentage_processor_time=CPU_LessThan30
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
operation=GetClientPendingServerEvents 831 conf:(1)

68. operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 831 conf:(1)

69. operation=GetClientPendingServerEvents
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
server_percentage_processor_time=CPU_LessThan30 831 conf:(1)

70. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30 831
conf:(1)

71. operation=GetClientPendingServerEvents
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
server_percentage_processor_time=CPU_LessThan30 831 conf:(1)

72. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
memory_available_gb=MemoryAvailableGB_Between7and8GB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
operation=GetClientPendingServerEvents 831 conf:(1)

73. operation=GetClientPendingServerEvents
memory_available_gb=MemoryAvailableGB_Between7and8GB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 831 conf:(1)

74. operation=GetClientPendingServerEvents
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 831 conf:(1)

75. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
operation=GetClientPendingServerEvents memory_available_gb=MemoryAvailableGB_Between7and8GB
831 conf:(1)

76. operation=GetClientPendingServerEvents
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>

responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
memory_available_gb=MemoryAvailableGB_Between7and8GB 831 conf:(1)

77. operation=GetClientPendingServerEvents
memory_available_gb=MemoryAvailableGB_Between7and8GB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
server_percentage_processor_time=CPU_LessThan30 831 conf:(1)

78. operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 831 conf:(1)

79. operation=GetClientPendingServerEvents
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 831 conf:(1)

80. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
memory_available_gb=MemoryAvailableGB_Between7and8GB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
server_percentage_processor_time=CPU_LessThan30 831 conf:(1)

81. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
server_percentage_processor_time=CPU_LessThan30
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 831 conf:(1)

82. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 831 conf:(1)

83. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
operation=GetClientPendingServerEvents 831 conf:(1)

84. operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 831 conf:(1)

85. operation=GetClientPendingServerEvents
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
memory_available_gb=MemoryAvailableGB_Between7and8GB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
server_percentage_processor_time=CPU_LessThan30 831 conf:(1)

86. operation=GetClientPendingServerEvents
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
server_percentage_processor_time=CPU_LessThan30
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 831 conf:(1)

87. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
memory_available_gb=MemoryAvailableGB_Between7and8GB

hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30 831
conf:(1)

88. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
server_percentage_processor_time=CPU_LessThan30
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
operation=GetClientPendingServerEvents memory_available_gb=MemoryAvailableGB_Between7and8GB
831 conf:(1)

89. operation=GetClientPendingServerEvents
memory_available_gb=MemoryAvailableGB_Between7and8GB
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
server_percentage_processor_time=CPU_LessThan30 831 conf:(1)

90. operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
memory_available_gb=MemoryAvailableGB_Between7and8GB 831 conf:(1)

91. operation=GetClientPendingServerEvents
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 831 conf:(1)

92. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 831 conf:(1)

93. operation=GetClientPendingServerEvents
hd_avg_disk_KB_transfer=HDAvrgKBTransfer_Between10and20KB 831 ==>
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 831 conf:(1)

94. operation=GetEvent 552 ==> server_percentage_processor_time=CPU_LessThan30 552 conf:(1)

95. operation=GetEvent 552 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 552
conf:(1)

96. operation=GetEvent memory_available_gb=MemoryAvailableGB_Between7and8GB 552 ==>
server_percentage_processor_time=CPU_LessThan30 552 conf:(1)

97. operation=GetEvent server_percentage_processor_time=CPU_LessThan30 552 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 552 conf:(1)

98. operation=GetEvent 552 ==> server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 552 conf:(1)

99. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==>
operation=GetClientPendingServerEvents 549 conf:(1)

100. operation=GetClientPendingServerEvents
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==>
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 549 conf:(1)

101. operation=GetClientPendingServerEvents
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==>
server_percentage_processor_time=CPU_LessThan30 549 conf:(1)

102. operation=GetClientPendingServerEvents
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 549 conf:(1)

103. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==>
server_percentage_processor_time=CPU_LessThan30 549 conf:(1)

104. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 549 conf:(1)

105. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
server_percentage_processor_time=CPU_LessThan30
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==>
operation=GetClientPendingServerEvents 549 conf:(1)

106. operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==>
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 549 conf:(1)

107. operation=GetClientPendingServerEvents
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==>
server_percentage_processor_time=CPU_LessThan30 549 conf:(1)

108. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==>
operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30 549
conf:(1)

109. operation=GetClientPendingServerEvents
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==>
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
server_percentage_processor_time=CPU_LessThan30 549 conf:(1)

110. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
memory_available_gb=MemoryAvailableGB_Between7and8GB
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==>
operation=GetClientPendingServerEvents 549 conf:(1)

111. operation=GetClientPendingServerEvents
memory_available_gb=MemoryAvailableGB_Between7and8GB
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==>
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 549 conf:(1)

112. operation=GetClientPendingServerEvents
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 549 conf:(1)

113. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==> operation=GetClientPendingServerEvents memory_available_gb=MemoryAvailableGB_Between7and8GB 549 conf:(1)

114. operation=GetClientPendingServerEvents network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==> responsetimeclassification=GetClientPendingServerEvents_LessThan1sec memory_available_gb=MemoryAvailableGB_Between7and8GB 549 conf:(1)

115. operation=GetClientPendingServerEvents memory_available_gb=MemoryAvailableGB_Between7and8GB network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==> server_percentage_processor_time=CPU_LessThan30 549 conf:(1)

116. operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30 network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 549 conf:(1)

117. operation=GetClientPendingServerEvents network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==> server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 549 conf:(1)

118. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec memory_available_gb=MemoryAvailableGB_Between7and8GB network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==> server_percentage_processor_time=CPU_LessThan30 549 conf:(1)

119. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec server_percentage_processor_time=CPU_LessThan30 network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 549 conf:(1)

120. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==> server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 549 conf:(1)

121. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==> operation=GetClientPendingServerEvents 549 conf:(1)

122. operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==> responsetimeclassification=GetClientPendingServerEvents_LessThan1sec 549 conf:(1)

123. operation=GetClientPendingServerEvents responsetimeclassification=GetClientPendingServerEvents_LessThan1sec memory_available_gb=MemoryAvailableGB_Between7and8GB network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==> server_percentage_processor_time=CPU_LessThan30 549 conf:(1)

124. operation=GetClientPendingServerEvents
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
server_percentage_processor_time=CPU_LessThan30
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 549 conf:(1)

125. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
memory_available_gb=MemoryAvailableGB_Between7and8GB
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==>
operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30 549
conf:(1)

126. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
server_percentage_processor_time=CPU_LessThan30
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==>
operation=GetClientPendingServerEvents memory_available_gb=MemoryAvailableGB_Between7and8GB
549 conf:(1)

127. operation=GetClientPendingServerEvents
memory_available_gb=MemoryAvailableGB_Between7and8GB
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==>
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
server_percentage_processor_time=CPU_LessThan30 549 conf:(1)

128. operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==>
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
memory_available_gb=MemoryAvailableGB_Between7and8GB 549 conf:(1)

129. operation=GetClientPendingServerEvents
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==>
server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 549 conf:(1)

130. responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==>
operation=GetClientPendingServerEvents server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 549 conf:(1)

131. operation=GetClientPendingServerEvents
network_KB_totalpersec=NetworkKBPerSec_Between40and50KB 549 ==>
responsetimeclassification=GetClientPendingServerEvents_LessThan1sec
server_percentage_processor_time=CPU_LessThan30
memory_available_gb=MemoryAvailableGB_Between7and8GB 549 conf:(1)

132. operation=GetEventHeader 545 ==> server_percentage_processor_time=CPU_LessThan30 545
conf:(1)

133. operation=GetEventHeader 545 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB
545 conf:(1)

134. operation=GetEventHeader memory_available_gb=MemoryAvailableGB_Between7and8GB 545 ==>
server_percentage_processor_time=CPU_LessThan30 545 conf:(1)

135. operation=GetEventHeader server_percentage_processor_time=CPU_LessThan30 545 ==>
memory_available_gb=MemoryAvailableGB_Between7and8GB 545 conf:(1)

136. operation=GetEventHeader 545 ==> server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 545 conf:(1)

137. responsetimeclassification=GetEvent_LessThan1sec 539 ==> operation=GetEvent 539 conf:(1)

138. responsetimeclassification=GetEvent_LessThan1sec 539 ==> server_percentage_processor_time=CPU_LessThan30 539 conf:(1)

139. responsetimeclassification=GetEvent_LessThan1sec 539 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 539 conf:(1)

140. responsetimeclassification=GetEvent_LessThan1sec server_percentage_processor_time=CPU_LessThan30 539 ==> operation=GetEvent 539 conf:(1)

141. operation=GetEvent responsetimeclassification=GetEvent_LessThan1sec 539 ==> server_percentage_processor_time=CPU_LessThan30 539 conf:(1)

142. responsetimeclassification=GetEvent_LessThan1sec 539 ==> operation=GetEvent server_percentage_processor_time=CPU_LessThan30 539 conf:(1)

143. responsetimeclassification=GetEvent_LessThan1sec memory_available_gb=MemoryAvailableGB_Between7and8GB 539 ==> operation=GetEvent 539 conf:(1)

144. operation=GetEvent responsetimeclassification=GetEvent_LessThan1sec 539 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 539 conf:(1)

145. responsetimeclassification=GetEvent_LessThan1sec 539 ==> operation=GetEvent memory_available_gb=MemoryAvailableGB_Between7and8GB 539 conf:(1)

146. responsetimeclassification=GetEvent_LessThan1sec memory_available_gb=MemoryAvailableGB_Between7and8GB 539 ==> server_percentage_processor_time=CPU_LessThan30 539 conf:(1)

147. responsetimeclassification=GetEvent_LessThan1sec server_percentage_processor_time=CPU_LessThan30 539 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 539 conf:(1)

148. responsetimeclassification=GetEvent_LessThan1sec 539 ==> server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 539 conf:(1)

149. responsetimeclassification=GetEvent_LessThan1sec server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 539 ==> operation=GetEvent 539 conf:(1)

150. operation=GetEvent responsetimeclassification=GetEvent_LessThan1sec memory_available_gb=MemoryAvailableGB_Between7and8GB 539 ==> server_percentage_processor_time=CPU_LessThan30 539 conf:(1)

151. operation=GetEvent responsetimeclassification=GetEvent_LessThan1sec server_percentage_processor_time=CPU_LessThan30 539 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 539 conf:(1)

152. responsetimeclassification=GetEvent_LessThan1sec memory_available_gb=MemoryAvailableGB_Between7and8GB 539 ==> operation=GetEvent server_percentage_processor_time=CPU_LessThan30 539 conf:(1)

153. responsetimeclassification=GetEvent_LessThan1sec server_percentage_processor_time=CPU_LessThan30 539 ==> operation=GetEvent memory_available_gb=MemoryAvailableGB_Between7and8GB 539 conf:(1)

154. operation=GetEvent responsetimeclassification=GetEvent_LessThan1sec 539 ==> server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 539 conf:(1)

155. responsetimeclassification=GetEvent_LessThan1sec 539 ==> operation=GetEvent server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 539 conf:(1)

156. operation=GetNotificationHistoryCurrentUser 538 ==> server_percentage_processor_time=CPU_LessThan30 538 conf:(1)

157. operation=GetNotificationHistoryCurrentUser 538 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 538 conf:(1)

158. operation=GetNotificationDetails 538 ==> server_percentage_processor_time=CPU_LessThan30 538 conf:(1)

159. operation=GetNotificationDetails 538 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 538 conf:(1)

160. operation=GetNotificationHistoryCurrentUser memory_available_gb=MemoryAvailableGB_Between7and8GB 538 ==> server_percentage_processor_time=CPU_LessThan30 538 conf:(1)

161. operation=GetNotificationHistoryCurrentUser server_percentage_processor_time=CPU_LessThan30 538 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 538 conf:(1)

162. operation=GetNotificationHistoryCurrentUser 538 ==> server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 538 conf:(1)

163. operation=GetNotificationDetails memory_available_gb=MemoryAvailableGB_Between7and8GB 538 ==> server_percentage_processor_time=CPU_LessThan30 538 conf:(1)

164. operation=GetNotificationDetails server_percentage_processor_time=CPU_LessThan30 538 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 538 conf:(1)

165. operation=GetNotificationDetails 538 ==> server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 538 conf:(1)

166. responsetimeclassification=GetNotificationHistoryCurrentUser_LessThan1sec 536 ==> operation=GetNotificationHistoryCurrentUser 536 conf:(1)

167. responsetimeclassification=GetNotificationHistoryCurrentUser_LessThan1sec 536 ==> server_percentage_processor_time=CPU_LessThan30 536 conf:(1)

168. responsetimeclassification=GetNotificationHistoryCurrentUser_LessThan1sec 536 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 536 conf:(1)

169. responsetimeclassification=GetNotificationHistoryCurrentUser_LessThan1sec server_percentage_processor_time=CPU_LessThan30 536 ==> operation=GetNotificationHistoryCurrentUser 536 conf:(1)

170. operation=GetNotificationHistoryCurrentUser responsetimeclassification=GetNotificationHistoryCurrentUser_LessThan1sec 536 ==> server_percentage_processor_time=CPU_LessThan30 536 conf:(1)

171. responsetimeclassification=GetNotificationHistoryCurrentUser_LessThan1sec 536 ==> operation=GetNotificationHistoryCurrentUser server_percentage_processor_time=CPU_LessThan30 536 conf:(1)

172. responsetimeclassification=GetNotificationHistoryCurrentUser_LessThan1sec memory_available_gb=MemoryAvailableGB_Between7and8GB 536 ==> operation=GetNotificationHistoryCurrentUser 536 conf:(1)

173. operation=GetNotificationHistoryCurrentUser responsetimeclassification=GetNotificationHistoryCurrentUser_LessThan1sec 536 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 536 conf:(1)

174. responsetimeclassification=GetNotificationHistoryCurrentUser_LessThan1sec 536 ==> operation=GetNotificationHistoryCurrentUser memory_available_gb=MemoryAvailableGB_Between7and8GB 536 conf:(1)

175. responsetimeclassification=GetNotificationHistoryCurrentUser_LessThan1sec memory_available_gb=MemoryAvailableGB_Between7and8GB 536 ==> server_percentage_processor_time=CPU_LessThan30 536 conf:(1)

176. responsetimeclassification=GetNotificationHistoryCurrentUser_LessThan1sec server_percentage_processor_time=CPU_LessThan30 536 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 536 conf:(1)

177. responsetimeclassification=GetNotificationHistoryCurrentUser_LessThan1sec 536 ==> server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 536 conf:(1)

178. responsetimeclassification=GetNotificationHistoryCurrentUser_LessThan1sec server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 536 ==> operation=GetNotificationHistoryCurrentUser 536 conf:(1)

179. operation=GetNotificationHistoryCurrentUser responsetimeclassification=GetNotificationHistoryCurrentUser_LessThan1sec memory_available_gb=MemoryAvailableGB_Between7and8GB 536 ==> server_percentage_processor_time=CPU_LessThan30 536 conf:(1)

180. operation=GetNotificationHistoryCurrentUser responsetimeclassification=GetNotificationHistoryCurrentUser_LessThan1sec server_percentage_processor_time=CPU_LessThan30 536 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 536 conf:(1)

181. responsetimeclassification=GetNotificationHistoryCurrentUser_LessThan1sec memory_available_gb=MemoryAvailableGB_Between7and8GB 536 ==> operation=GetNotificationHistoryCurrentUser server_percentage_processor_time=CPU_LessThan30 536 conf:(1)

182. responsetimeclassification=GetNotificationHistoryCurrentUser_LessThan1sec server_percentage_processor_time=CPU_LessThan30 536 ==> operation=GetNotificationHistoryCurrentUser memory_available_gb=MemoryAvailableGB_Between7and8GB 536 conf:(1)

183. operation=GetNotificationHistoryCurrentUser responsetimeclassification=GetNotificationHistoryCurrentUser_LessThan1sec 536 ==> server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 536 conf:(1)

184. responsetimeclassification=GetNotificationHistoryCurrentUser_LessThan1sec 536 ==> operation=GetNotificationHistoryCurrentUser server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 536 conf:(1)

185. responsetimeclassification=GetNotificationDetails_LessThan1sec 529 ==> operation=GetNotificationDetails 529 conf:(1)

186. responsetimeclassification=GetNotificationDetails_LessThan1sec 529 ==> server_percentage_processor_time=CPU_LessThan30 529 conf:(1)

187. responsetimeclassification=GetNotificationDetails_LessThan1sec 529 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 529 conf:(1)

188. responsetimeclassification=GetNotificationDetails_LessThan1sec server_percentage_processor_time=CPU_LessThan30 529 ==> operation=GetNotificationDetails 529 conf:(1)

189. operation=GetNotificationDetails responsetimeclassification=GetNotificationDetails_LessThan1sec 529 ==> server_percentage_processor_time=CPU_LessThan30 529 conf:(1)

190. responsetimeclassification=GetNotificationDetails_LessThan1sec 529 ==> operation=GetNotificationDetails server_percentage_processor_time=CPU_LessThan30 529 conf:(1)

191. responsetimeclassification=GetNotificationDetails_LessThan1sec memory_available_gb=MemoryAvailableGB_Between7and8GB 529 ==> operation=GetNotificationDetails 529 conf:(1)

192. operation=GetNotificationDetails responsetimeclassification=GetNotificationDetails_LessThan1sec 529 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 529 conf:(1)

193. responsetimeclassification=GetNotificationDetails_LessThan1sec 529 ==> operation=GetNotificationDetails memory_available_gb=MemoryAvailableGB_Between7and8GB 529 conf:(1)

194. responsetimeclassification=GetNotificationDetails_LessThan1sec memory_available_gb=MemoryAvailableGB_Between7and8GB 529 ==> server_percentage_processor_time=CPU_LessThan30 529 conf:(1)

195. responsetimeclassification=GetNotificationDetails_LessThan1sec server_percentage_processor_time=CPU_LessThan30 529 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 529 conf:(1)

196. responsetimeclassification=GetNotificationDetails_LessThan1sec 529 ==> server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 529 conf:(1)

197. responsetimeclassification=GetNotificationDetails_LessThan1sec server_percentage_processor_time=CPU_LessThan30 memory_available_gb=MemoryAvailableGB_Between7and8GB 529 ==> operation=GetNotificationDetails 529 conf:(1)

198. operation=GetNotificationDetails responsetimeclassification=GetNotificationDetails_LessThan1sec memory_available_gb=MemoryAvailableGB_Between7and8GB 529 ==> server_percentage_processor_time=CPU_LessThan30 529 conf:(1)

199. operation=GetNotificationDetails responsetimeclassification=GetNotificationDetails_LessThan1sec server_percentage_processor_time=CPU_LessThan30 529 ==> memory_available_gb=MemoryAvailableGB_Between7and8GB 529 conf:(1)

200. responsetimeclassification=GetNotificationDetails_LessThan1sec memory_available_gb=MemoryAvailableGB_Between7and8GB 529 ==> operation=GetNotificationDetails server_percentage_processor_time=CPU_LessThan30 529 conf:(1)