**Virtual Machine Performance Benchmark Tool**

**Manual del Programador**

**Carlos Eduardo Gómez Montoya**1,2**, MSc, PhD(c).**

**Cristian David Sierra Barrera**1**, Ing.**

**Harold Enrique Castro Barrera**1**, PhD.**

1**Universidad de Los Andes**

2**Universidad del Quindío**

**2017**

**CONTENIDO**

1 INTRODUCCIÓN 1

2 VISTA GENERAL DE LA SOLUCIÓN 2

3 ATRIBUTOS DE CALIDAD 3

4 DIAGRAMA DE CLASES 5

4.1 CLASES PRINCIPALES 5

4.1.1 AdminServer 5

4.1.2 Benchmarking 5

4.1.3 Configuration 5

4.1.4 LoadTestSpecification 6

4.1.5 Scenario 6

4.1.6 HardwareProfile 6

4.1.7 VirtualMachine 6

4.1.8 VirtualBoxVM 6

4.2 OTRAS CLASES 6

4.2.1 Constants 6

4.2.2 LoggerUtil 6

4.2.3 Services 7

5 DIAGRAMA DE SECUENCIA 8

6 DIAGRAMA DE DESPLIEGUE 9

7 DIAGRAMA DE PAQUETES 10

8 JAVADOC 11

# INTRODUCCIÓN

La computación en la nube ha venido tomando cada vez más fuerza en distintos ambientes, impactando todos los sectores de la economía. La virtualización es una de las tecnologías que han permitido que la computación en la nube pueda ser llevada a la realidad sin importar las dimensiones del usuario que requiere sus servicios.

El análisis de la ejecución simultánea de máquinas virtuales es esencial para la asignación de recursos computacionales en la computación en la nube. La herramienta de software Virtual Machine Performance Benchmark Tool se ha desarrollado con el fin de apoyar el desarrollo de la tesis doctoral “Confiabilidad en sistemas desktop cloud” que actualmente se encuentra en desarrollo por el estudiante doctoral Carlos Eduardo Gómez Montoya en la Universidad de Los Andes. Este programa es una aplicación de software creada para establecer la configuración de máquinas virtuales y definir distintas aplicaciones de carga, de tal forma que se pueda evaluar un método de benchmarking elegido por el usuario, entre los disponibles en el sistema. Dado que realizar estas pruebas generalmente es un trabajo repetitivo y tedioso, la principal funcionalidad consiste en la automatización de escenarios y máquinas virtuales de acuerdo con parámetros que son ajustados por el usuario a través de un archivo de texto en formato JSON con una sintaxis específica.

Esta herramienta de software es de gran utilidad porque no solo se trata de la ejecución automática de despliegues de software sino que además entrega los resultados en forma de valores separados por coma (.csv), los cuales pueden ser analizados en una hoja de Excel o en cualquier programa de análisis estadísticos que reciba datos en este formato.

Se trata de una aplicación cliente – servidor en la cual el servidor corre en el host mientras que el cliente corre en la máquina virtual. El servidor enciende la máquina virtual, mide el tiempo que se tarda la máquina virtual en estar lista para operar y dependiendo de los parámetros de configuración especificados por el usuario, ejecuta y termina las aplicaciones de benchmarking y apaga la máquina virtual. De igual manera, mientras las máquinas virtuales estén creadas, el servidor puede modificar sus especificaciones, otra vez, de acuerdo con los parámetros de entrada especificados.

Este manual incluye una vista general de la solución, atributos de calidad incluidos en el desarrollo de la aplicación y los diagramas de clases, de secuencia, de despliegue y de paquetes, además de la documentación en formato Javadoc para la referencia de los programadores.

# VISTA GENERAL DE LA SOLUCIÓN

La solución está conformada por una aplicación tipo servidor que se ejecuta en un host y una aplicación cliente que se ejecuta en todas las máquinas virtuales que corren en el host. Entre ellas conforman una red en modo Host-Only, es decir, una red aislada que no tiene comunicación con el mundo exterior.

Las máquinas virtuales están configuradas de tal manera que inician automáticamente y se ejecuta la aplicación cliente sin la intervención del usuario. Al iniciar su ejecución, luego del proceso de arranque del sistema, envían un mensaje al servidor para que se registre el tiempo que tardó el proceso de arranque del sistema operativo en la máquina virtual. A continuación, el servidor da una orden para iniciar una prueba de carga en el cliente. Esta prueba de carga estará ejecutándose de manera permanente hasta que el servidor ordene que se apague la máquina virtual. En el servidor, luego de ajustar las propiedades de cada máquina virtual que sean establecidas en el archivo de configuración y ponerlas en ejecución, ejecuta una prueba de carga que será empleada como caso de prueba para medir el efecto que tiene la ejecución simultánea de máquinas virtuales en un host.

Vale la pena aclarar que en el archivo de configuración se puede determinar el número de máquinas virtuales a ejecutar y los perfiles de hardware para cada una de ellas. En la figura a continuación, a manera de ejemplo, se puede ver el host y la red que lo comunica con tres máquinas virtuales.

**Figura 1. Vista general de la solución**



# ATRIBUTOS DE CALIDAD

A continuación se presentan brevemente los atributos de calidad tenidos en cuenta durante el diseño de la aplicación.

**Extensibilidad:** Como mecanismo para facilitar la extensibilidad dentro del sistema se utilizaron el uso de patrones de diseño que fomentaran dicho atributo, como son el uso de la herencia para la incorporación de nuevos hipervisores, el bajo acoplamiento como en el caso del manejo de archivos de configuración para no tener que modificar el código de la aplicación si se necesita cambiar algún parámetro. Igualmente, el diseño de la aplicación está pensado para que se puedan agregar componentes para incrementar su funcionalidad.

**Desempeño:** Por tratarse de una aplicación para medir el desempeño, esta herramienta de software busca encontrar distintos comportamientos en esta materia, logrando en ocasiones que la máquina física sienta realmente que la ejecución simultánea de máquinas virtuales afecta su rendimiento.

**Usabilidad:** La usabilidad de esta aplicación depende de archivos de configuración para facilitar la interacción con el usuario. Otros aspectos en este sentido no fueron tenidos en cuenta, dada la naturaleza de la aplicación.

**Integridad:** En este sentido se utiliza control de excepciones y manejo de archivos log para el registro de los eventos del sistema y la salida de los mensajes de depuración.

**Portabilidad:** El software fue desarrollado en Java y se utilizaron librerías estándar, por lo que el servidor puede correr sin modificaciones en cualquier sistema operativo. Para las pruebas de carga se utilizaron programas de benchmarking ampliamente conocidos para su fácil ejecución en cualquier distribución de sistemas GNU/Linux. Estas aplicaciones de benchmarking serán utilizadas para la ejecución del cliente en las máquinas virtuales.

**Compatibilidad:** No aplica, ya que se trata de la primera versión del software.

**Mantenimiento:** El uso de un IDE estándar como Eclipse y un lenguaje ampliamente conocido como Java junto con el estándar de codificación para el nombramiento de identificadores, constantes, variables, métodos, clases y objetos facilita el mantenimiento del sistema por parte del mismo grupo de desarrollo o de otras personas si se requiere, especialmente si se encuentran defectos que puedan ser fácilmente identificables y se puedan corregir.

**Documentación:** Para la documentación del software desarrollado se hizo un proceso de ingeniería de software ligero con levantamiento de requerimientos a partir de las historias de usuario, y se escribieron los comentarios para la generación de la documentación en formato Javadoc.

# DIAGRAMA DE CLASES

A continuación se describe el estado de las clases utilizadas, indicando sus relaciones y los tipos de clases usados para lograr el funcionamiento de la aplicación.

/Users/Cristian/Downloads/Diagrama de Clases.png

A continuación se describen las clases empleadas en la implementación de la solución:

## CLASES PRINCIPALES

### AdminServer

Esta es la clase principal de la aplicación. Realiza las ejecuciones con base en los datos cargados por el archivo de configuración. Se encarga de imprimir los resultados de las mediciones.

### Benchmarking

Clase estática encargada de mantener las aplicaciones de benchmarking disponibles para la configuración.

### Configuration

Esta clase permite cargar las propiedades mediante las cuales se configura el servidor. Tiene como uso la asignación de datos importantes para el funcionamiento del servidor. En particular, la ruta del hipervisor y dónde residen las máquinas virtuales son parámetros fundamentales para la ejecución de las pruebas.

### LoadTestSpecification

Como su nombre lo indica, se encarga de hacer la lectura de los parámetros configurables y los ajusta en el AdminServer para realizar la ejecución.

### Scenario

Esta clase está encargada de representar un escenario en la aplicación. Note que posee el número de evaluaciones a realizar además de los perfiles que compenen cada escenario.

### HardwareProfile

Esta clase representa el perfil de hardware que pueden tener las máquinas virtuales, tiene como atributos el nombre, la cantidad de RAM en Gigabytes y el número de núcleos de procesamiento (cores) a utilizar.

### VirtualMachine

Es una clase abstracta que representa una máquina virtual. Corresponde a una clase abstacta para facilitar el desarrollo con un hipervisor específico y permitir que la aplicación pueda funcionar correctamente con distintos hipervisores.

### VirtualBoxVM

Esta clase implementa la clase VirtualMachine, ajustando los métodos con los comandos específicos del hipervisor VirtualBox.

## OTRAS CLASES

### Constants

Clase estática que mantiene las constantes del proyecto.

### LoggerUtil

Clase encargada de generar los Logs de ejecución.

### Services

Se encarga de ejecutar funcionalidades del sistema.

# DIAGRAMA DE SECUENCIA

Uno de los procesos más importantes es la comunicación entre servidor y cliente para hacer las mediciones. El siguiente diagrama describe el proceso de comunicación entre el servidor y las máquinas cliente.

Diagrama%20de%20Secuencia.png

El procedimiento simplificado consiste en encender las máquinas esperar la conexión, enviar la configuración deseada. Cuando todas las máquinas hayan sido configuradas se inicia el procedimiento de medición. Finalmente, cuando es acabado el procedimiento todas la máquinas son apagadas, con ello el componente TCPClient que está contenido en cada máquina virtual.

# DIAGRAMA DE DESPLIEGUE

El siguiente diagrama ilustra cómo se ubican los componentes de la aplicación en los distintos artefactos que componen la arquitectura.

Diagrama%20de%20Despliegue.png

En primer lugar, se debe señalar que la aplicación tiene dos componentes principales: el *AdminServer* y el *TCPClient*. El primero realiza las mediciones y ejecuciones de escenarios. El segundo se encarga de ejecutar pruebas de carga en las máquinas virtuales seleccionadas.

En segundo lugar, note que la arquitectura es extensible en términos de máquinas virtuales. Sin embargo, sólo responden a un único servidor. Se debe ser cuidadoso al crear una red que permita la conexión entre servidor y cliente, puesto que, de esto depende la variedad de pruebas que se pueden realizar.

# DIAGRAMA DE PAQUETES

Las clases y paquetes que componen la arquitectura de la aplicación está descrito por el siguiente diagrama. Las dependencias entre paquetes pueden ser vistas, de tal forma que sea más fácil la implementación de nuevas funcionalidades a la aplicación.

/Users/Cristian/Downloads/Diagrama de Paquetes.png

# JAVADOC

El documento de documentación de Java (Javadoc) se presenta a continuación para referencia de los programadores.