Análisis de Rendimiento para Soluciones de Cloud Computing

Marcelo Damián Parrino [1]

[1]Facultad de Ingeniería, Universidad de Palermo
Mario Bravo 1050, Buenos Aires, Argentina
mparri@palermo.edu

Abstract—*Este trabajo presenta un conjunto de algoritmos de benchmarking, junto con un Framework asociado con el fin de evaluar objetivamente el rendimiento computacional de diferentes soluciones de "Cloud Computing". Los resultados obtenidos por las diferentes pruebas dentro del Framework permitieron analizar la performance de procesamiento computacional de cada solución, posibilitando su comparación entre sí, y determinando finalmente la mejor solución de "Cloud Computing" disponible en el mercado.*

Index Terms—*benchmark, cloud computing, Dhrystone, Whetstone, Linpack, Amazon EC2, Microsoft Azure, Google App Engine*

1. Introduccion

Se investigaron algunos de los posibles algoritmos aplicables al análisis de rendimiento de diferentes soluciones actualmente disponibles de "Cloud Computing", para poder evaluar entonces el rendimiento comparado de cada una de ellas.

La idea fundamental fue seleccionar el conjunto de algoritmos más adecuado, ajustarlo y probarlo para evaluar de una forma objetiva las soluciones de "Computación en la Nube" de diferentes proveedores del mercado, logrando de esa forma una visión más clara y objetiva de las capacidades de computo provistas por cada uno de ellos.

El objetivo de este trabajo consistió en desarrollar una investigación basada en un relevamiento bibliográfico, plantear un conjunto de pruebas objetivas de rendimiento y performance para las "Nubes Computacionales" y escribir un informe técnico aplicando esas pruebas sobre las soluciones comerciales disponibles de "Cloud Computing".

El primer objetivo específico fue investigar puntualmente algunos de los algoritmos disponibles junto con sus posibles optimizaciones y mejoras, que permitieran obtener datos objetivos para el análisis de rendimiento de las soluciones de "Cloud Computing".

Como segundo objetivo específico, a partir de los algoritmos y modelos analizados en la etapa previa, se desarrolló, utilizando los SDK disponibles, una aplicación de software que implementó las pruebas de rendimiento planteadas, registrando los resultados para su posterior análisis y comparación.

El último objetivo fue la evaluación y análisis de los resultados obtenidos, a partir de la implementación de las pruebas de software. La evaluación de los resultados permitió analizar el desempeño y rendimiento objetivos de las diferentes soluciones disponibles de "Cloud Computing".

2. Benchmarks

2.1. Whetstone

Whetstone es un benchmark científico diseñado en el Laboratorio Nacional de Física de Inglaterra. Se lo considera uno de los padres de los benchmarks sintéticos por ser el primero diseñado específicamente con ese fin. Hoy en día forma parte de muchos benchmarks actuales.

El programa fue diseñado basándose en el estudio de 949 programas Algol '60, por lo que constituye un benchmark "sintético". Debido a que sus creadores se basaron en programas científicos de la época, este benchmark puede considerarse como científico. Fue diseñado para medir la velocidad de ejecución de una variedad de instrucciones de coma flotante (+, *, seno, cos, atan, sqrt, log, exp) en datos escalares o vectoriales, aunque también contiene algo de código de enteros, condicionales y llamadas a procedimientos. En líneas generales intenta estimar la velocidad de la CPU con la FPU.

En el benchmark Whetstone, el código objeto que se itera es muy pequeño, cabiendo completamente en la caché interna de los procesadores modernos, por lo tanto manteniendo el pipeline de la FPU completo y a la FPU permanentemente ocupada. Esto es deseable porque el Whetstone hace exactamente lo que nosotros queremos que haga: medir de la FPU, no el acoplamiento de la CPU (caché externa/memoria principal), rendimiento de enteros o cualquier otra característica del sistema bajo prueba. El número de iteraciones es lo suficientemente grande como para que el costo por instrucciones extras sea despreciable.

2.2. Dhrystone

Dhrystone es un benchmark sintético que pretende ser representativo de la programación entera de sistemas. Está basado en estadísticas publicadas sobre uso de

particularidades de los lenguajes de programación, sistemas operativos, compiladores, editores, etc.

Fue diseñado también basándose en programas existentes (a partir de una mezcla representativa), pero los datos eran de programas comerciales en lugar de científicos. Uno de sus usos actuales es como parte de otros benchmarks. Es un benchmark sintético y comercial.

Dhrystone intenta medir la velocidad del sistema en cuanto a rendimiento no numérico, expresando los resultados en DPS (instrucciones Dhrystones Por Segundo).

Dhrystone contiene muchas instrucciones simples, llamadas a procedimiento y condicionales, y pocas de coma flotante y bucles. No realiza llamadas al sistema. Usa pocas variables globales y ejecuta operaciones con punteros. Está compuesto por 12 procedimientos incluidos en un bucle de medida con 94 sentencias. No se puede variar su tamaño. Está compuesto por un 53% de instrucciones de asignación, 32% de instrucciones de control y un 15% de llamadas a procedimiento.

Por ser pequeño, el benchmark Dhrystone entra completamente en la caché interna, de esta manera no mide el resto del sistema pero presenta la ventaja de que mide solamente la capacidad del procesador para trabajar con enteros.

2.3. Linpack

El benchmark Linpack fue desarrollado en el Argone National Laboratory por Jack Dongarra en 1976, y es uno de los más usados en sistemas científicos y de ingeniería. Su uso como benchmark fue accidental, ya que originalmente fue una extensión del programa Linpack -cuyo propósito era resolver sistemas de ecuaciones- que otorgaba el tiempo de ejecución del programa en 23 máquinas distintas.

La característica principal de Linpack es que hacen un uso muy intensivo de las operaciones de coma flotante, por lo que sus resultados son muy dependientes de la capacidad de la FPU que tenga el sistema. El benchmark pasa la mayor parte del tiempo ejecutando unas rutinas llamadas BLAS (Basic Linear Algebra Subroutines o Subrutinas de Álgebra Lineal Básica).

El mayor tiempo de ejecución se consume en la rutina DAXPY de la biblioteca BLAS (casi el 90%). DAXPY realiza el siguiente cálculo:

$$y(i) := y(i) + a * x(i).$$

Al realizar esencialmente cálculos con matrices es un test fácilmente paralelizable, y se puede utilizar para medir la eficiencia de sistemas multiprocesador.

El reporte del benchmark describe la performance para resolver un problema de matrices generales densas $Ax = b$ a tres niveles de tamaño y oportunidad de optimización.

3. Desarrollo de las pruebas

3.1. Diseño del Framework

Como los benchmarks elegidos son intrínsecamente diferentes y evalúan diferentes aspectos de un sistema computacional, se decidió desarrollar un marco o "Framework" uniforme que permita homogeneizar diferentes tipos de pruebas, estandarizando la ejecución de ciclos de pruebas, la contabilización de tiempos de ejecución y la recolección y presentación de resultados.

Así mismo se intentó plantear un modelo suficientemente simple para que pudiera ser implementado simultáneamente en diferentes lenguajes de programación sin necesidad de rediseñar el Framework.

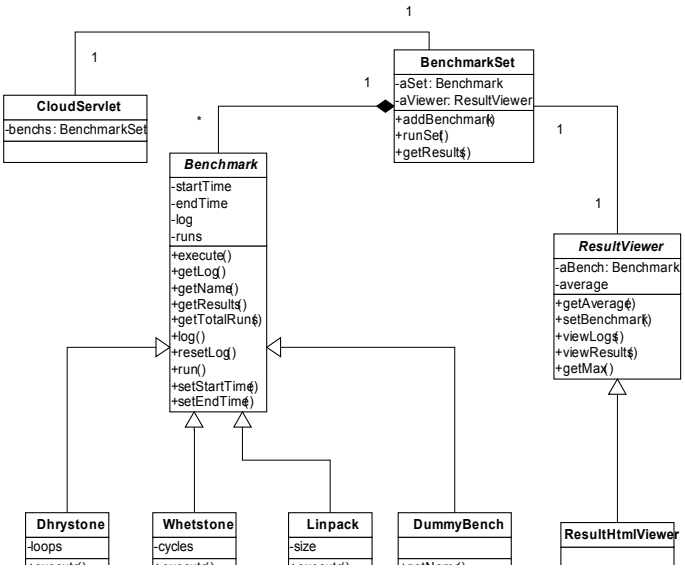

Fig. 1. Diagrama UML "Benchmark Framework"

En la figura 1 se presenta el diagrama UML del Framework propuesto para el problema de ejecución de Benchmarks en la nube.

A continuación se describen las clases más importantes del modelo:

- «Benchmark»: clase abstracta de la cual deben heredar todos los benchmarks que se deseen incorporar al modelo. Se encarga fundamentalmente de ejecutar los bucles de código que conforman la prueba y contabilizar los tiempos de ejecución.
- «ResultViewer»: clase abstracta encargada manejar y dar formato a los resultados generados por la corrida de un Benchmark.
- «ResultHtmlViewer»: implementación de ResultViewer que genera código HTML para ser presentado en la Web.
- «BenchmarkSet»: representa un conjunto o lote de benchmarks que serán ejecutados consecutivamente.
- «CloudServlet»: se encarga de interactuar con el servidor Web, recibir el pedido del usuario y lanzar

un BenchmarkSet, devolviendo posteriormente los resultados en formato HTML.

- «DummyBench»: es una prueba ficticia que consolida los resultados de todas las pruebas anteriores que se hayan ejecutado en el BenchmarkSet.

Se decidió realizar dos implementaciones del Framework: una en lenguaje Java que podrá ser utilizada tanto en Google App Engine como en Amazon EC2, y otra en lenguaje C# que será utilizada en Microsoft Azure.

Para las implementaciones de los algoritmos Whetstone, Dhrystone y Linpack, se buscó en Internet versiones ya implementadas en lenguaje Java, que luego se adaptaron a la estructura del Framework. Para evitar discrepancias en los resultados de los benchmarks, las implementaciones de Whetstone, Dhrystone y Linpack en C# se codificaron adaptando las versiones Java usadas anteriormente. De esta forma, cualquier optimización existente a nivel de código en los benchmarks estará presente en ambas implementaciones (Java y C#), minimizando las diferencias de los resultados por cuestiones de implementación.

3.2. Desarrollo en Google App Engine

El SDK de Google App Engine cuenta con un plugin para el entorno de desarrollo Eclipse, permitiendo crear, probar y subir aplicaciones de App Engine desde el propio IDE. El SDK también incluye herramientas de línea de comandos para ejecutar el servidor de desarrollo y subir la aplicación a la nube de Google.

Para el desarrollo e implementación del Framework de pruebas en Google App Engine, se utilizó el ambiente de desarrollo integrado de Eclipse, junto con el plugin y SDK para Java de Google, corriendo sobre Windows XP Profesional SP3.

Además, se creó una cuenta gratuita en App Engine [https://appengine.google.com/] para realizar las pruebas y la publicación de la aplicación en la nube.

Aunque Google App Engine también ofrece un SDK para Python, se decidió utilizar el lenguaje Java con el objetivo de estandarizar lo más posible las pruebas frente a las otras plataformas de Amazon y Microsoft.

Una vez desarrollada la codificación y testing en Java para App Engine, se utilizó el plugin de Eclipse para realizar la publicación del Servlet en la nube de Google [http://dparrino.appspot.com/aetest] directamente desde el propio entorno de desarrollo, dejando disponible la aplicación para su posterior ejecución.

3.3. Desarrollo en Microsoft Azure

Para el desarrollo e implementación del Framework de pruebas en Microsoft Azure se utilizó el ambiente de desarrollo integrado de Visual Web Developer 2008 Express Edition SP1 y SQL Server 2008 Express Edition, junto con el SDK de Microsoft, corriendo sobre Windows Vista Ultimate SP1.

Dado que el sistema operativo de escritorio era Windows XP y los requerimientos mínimos del SDK solicitaban Windows Vista, se utilizó la herramienta gratuita de virtualización Microsoft Virtual PC 2007 para poder ejecutar Windows Vista sobre el sistema operativo anfitrión Windows XP.

Además, se creó una cuenta gratuita en Microsoft Windows Azure [http://www.microsoft.com/windowsazure/] para realizar las pruebas y la posterior publicación de la aplicación en la nube.

Aunque el SDK de Azure provee soporte para varios lenguajes .NET, se decidió utilizar el lenguaje C# (dada su similitud con Java) con el objetivo de estandarizar dentro de lo posible las pruebas, frente a las otras plataformas de Amazon y Google.

Una vez desarrollada la codificación y testing en C# para Windows Azure, se utilizó el empaquetador incorporado al Visual Studio Express para generar el paquete binario para su publicación en la nube. Luego se publicó el paquete previamente generado en la nube de Microsoft [http://dparrino.cloudapp.net/] por medio de un navegador Web, dejando disponible la aplicación para su posterior ejecución.

3.4. Desarrollo en Amazon EC2

Para el desarrollo e implementación del Framework de pruebas en Amazon EC2 se utilizó el ambiente de desarrollo integrado de Eclipse, junto con el AWS Toolkit para Java de Amazon, corriendo sobre Windows XP Profesional SP3.

Además, se creó una cuenta en Amazon Web Services [http://aws.amazon.com/ec2/] para realizar las pruebas y la publicación de la aplicación en la nube.

Dado que Amazon EC2 ofrece virtualización de plataformas, se decidió utilizar una Java Web Starter AMI ("Amazon Machine Image") provista por AWS. La Java Web Starter AMI (ami-45e7002c) incluye Fedora Core 8 de 32 bits, Java JDK 7, Tomcat 5.5, Apache 2.2.9 y MySQL 5.0.

La imagen seleccionada permitió entonces utilizar el lenguaje Java para el desarrollo del Framework, con el objetivo de estandarizar lo más posible las pruebas frente a las otras plataformas de Google y Microsoft.

Una vez desarrollada la codificación y testing en Java para Amazon EC2, se generó desde Eclipse el paquete WAR para realizar la posterior publicación del Servlet en la instancia de Tomcat que se ejecuta en la nube de Amazon EC2.

Finalmente, autenticándose en el administrador del servidor Tomcat por medio de un navegador Web, se publicó el paquete WAR que contenía el Servlet, dejando disponible la aplicación para su posterior ejecución.

4. Ejecución de las Pruebas

A continuación se detallan los resultados de las pruebas ejecutadas en cada una de las tres soluciones de Cloud Computing que se eligieron para el análisis de rendimiento.

4.1. Ejecución en Google App Engine

Se realizaron 100 ejecuciones en Google App Engine de los tres benchmarks codificados en Java, con los siguientes criterios:

- Whetstone, 50 ciclos.
- Dhrystone, 1000000 bucles.
- Linpack, N=500.

Asimismo, la cuenta gratuita de App Engine define las siguientes restricciones:

- 6.5 horas de CPU al día.
- 1300000 solicitudes al día.
- 1 GB de tráfico de datos diario.

En la siguiente tabla se presentan un resumen de los resultados obtenidos al correr el benchmark Whetstone en Google App Engine:

TABLA I
Resultados Whetstone

Google App Engine: Whetstone	
Ejecuciones	100
Tiempo Total	30329 msegs
Tiempo Promedio	303,29 msegs
Tiempo Máximo	369 msegs

En la siguiente tabla se presentan un resumen de los resultados obtenidos al correr el benchmark Dhrystone en Google App Engine:

TABLA II
Resultados Dhrystone

Google App Engine: Dhrystone	
Ejecuciones	100
Tiempo Total	13557 msegs
Tiempo Promedio	135,57 msegs
Tiempo Máximo	170 msegs

En la siguiente tabla se presentan un resumen de los resultados obtenidos al correr el benchmark Linpack en Google App Engine:

TABLA III
Resultados Linpack

Google App Engine: Linpack	
Ejecuciones	100
Tiempo Total	19249 msegs
Tiempo Promedio	192,49 msegs
Tiempo Máximo	246 msegs

4.2. Ejecución en Microsoft Azure

Se realizaron 100 ejecuciones en Microsoft Azure de los tres benchmarks codificados en C#, con los siguientes criterios:

- Whetstone, 50 ciclos.
- Dhrystone, 1000000 bucles.
- Linpack, N=500.

Cabe destacar que las cuotas de uso diarias de Windows Azure para el "Technology Preview" no están informadas.

En la siguiente tabla se presentan un resumen de los resultados obtenidos al correr el benchmark Whetstone en Microsoft Azure:

TABLA IV
Resultados Whetstone

Microsoft Azure: Whetstone	
Ejecuciones	100
Tiempo Total	88602 msegs
Tiempo Promedio	886,02 msegs
Tiempo Máximo	999 msegs

En la siguiente tabla se presentan un resumen de los resultados obtenidos al correr el benchmark Dhrystone en Microsoft Azure:

TABLA V
Resultados Dhrystone

Microsoft Azure: Dhrystone	
Ejecuciones	100
Tiempo Total	80412 msegs
Tiempo Promedio	804,12 msegs
Tiempo Máximo	890 msegs

En la siguiente tabla se presentan un resumen de los resultados obtenidos al correr el benchmark Linpack en Microsoft Azure:

TABLA VI
Resultados Linpack

Microsoft Azure: Linpack	
Ejecuciones	100
Tiempo Total	17328 msegs
Tiempo Promedio	173,28 msegs
Tiempo Máximo	296 msegs

4.3. Ejecución en Amazon EC2

Se realizaron 100 ejecuciones en Amazon EC2 de los tres benchmarks codificados en Java, con los siguientes criterios:

- Whetstone, 50 ciclos.
- Dhrystone, 1000000 bucles.

- Linpack, N=500.

Asimismo, la AMI "Java Web Starter" se ejecutó como una instancia estándar ("Small Instance" por defecto), con las siguientes características:

- Plataforma de 32-bit
- 1.7 GB de memoria
- 1 unidad de computación EC2 (1 núcleo virtual con 1 unidad EC2)
- 160 GB de almacenamiento de instancia

En la siguiente tabla se presentan un resumen de los resultados obtenidos al correr el benchmark Whetstone en Amazon EC2:

TABLA VII
RESULTADOS WHETSTONE

Amazon EC2: Whetstone	
Ejecuciones	100
Tiempo Total	60590 msegs
Tiempo Promedio	605,9 msegs
Tiempo Máximo	713 msegs

En la siguiente tabla se presentan un resumen de los resultados obtenidos al correr el benchmark Dhrystone en Amazon EC2:

TABLA VIII
RESULTADOS DHRYSTONE

Amazon EC2: Dhrystone	
Ejecuciones	100
Tiempo Total	30143 msegs
Tiempo Promedio	301,43 msegs
Tiempo Máximo	349 msegs

En la siguiente tabla se presentan un resumen de los resultados obtenidos al correr el benchmark Linpack en Amazon EC2:

TABLA IX
RESULTADOS LINPACK

Amazon EC2: Linpack	
Ejecuciones	100
Tiempo Total	41280 msegs
Tiempo Promedio	412,8 msegs
Tiempo Máximo	682 msegs

5. EVALUACIÓN Y ANÁLISIS

En este capítulo se comparan y evalúan los resultados obtenidos en las diferentes ejecuciones, para analizar posteriormente cual es la solución de mejor rendimiento.

5.1. Evaluación de las pruebas

En la siguiente tabla se presenta el detalle completo los resultados obtenidos al correr el benchmark Whetstone en las tres plataformas:

TABLA X
RESULTADOS COMPARADOS WHETSTONE

Tiempo	**Google App Engine**	**Microsoft Azure**	**Amazon EC2**
Total	30329	88602	60590
Promedio	303,29	886,02	605,9
Máximo	369	999	713

Para la prueba Whetstone, Google App Engine tuvo el mejor rendimiento tanto en promedio como en menor tiempo máximo, seguido por Amazon EC2 (doble de tiempo) y Microsoft Azure (casi 3 veces más).

En la siguiente tabla se presenta el detalle completo los resultados obtenidos al correr el benchmark Dhrystone en las tres plataformas:

TABLA XI
RESULTADOS COMPARADOS DHRYSTONE

Tiempo	**Google App Engine**	**Microsoft Azure**	**Amazon EC2**
Total	13557	80412	30143
Promedio	135,57	804,12	301,43
Máximo	170	890	349

Para la prueba Dhrystone, Google App Engine tuvo nuevamente el mejor rendimiento tanto en promedio como en menor tiempo máximo, seguido por Amazon EC2 (doble de tiempo) y Microsoft Azure (casi 6 veces más).

Por último, en la siguiente tabla se presenta el detalle completo los resultados obtenidos al correr el benchmark Linpack en las tres plataformas:

TABLA XII
RESULTADOS COMPARADOS LINPACK

Tiempo	**Google App Engine**	**Microsoft Azure**	**Amazon EC2**
Total	19249	17328	41280
Promedio	192,49	173,28	412,8
Máximo	246	296	682

Para la prueba Linpack, Microsoft Azure tuvo el mejor rendimiento promedio, mientras que Google App Engine obtuvo el primer lugar en menor tiempo máximo. Amazon EC2 quedó ultimo (por más del doble de tiempo) en ambas métricas.

5.2. Análisis de los resultados

Para analizar de manera visual los resultados obtenidos en cada uno de los benchmarks, se generaron gráficos a partir de las tablas de tiempos de ejecución recolectadas.

Tal como se puede apreciar en las figuras 9, 10 y 11, el rendimiento de Google App Engine fue claramente superior a las otras dos alternativas (Amazon EC2 y Microsoft Azure) en Whetstone y Dhrystone, mientras que en el benchmark Linpack Google App Engine fue mínimamente superado en ocasiones por Microsoft Azure.

Cabe resaltar que tanto Azure como App Engine se ejecutaron sobre cuentas gratuitas, mientras que Amazon facturó las horas de CPU utilizadas y de todas formas no salió primero en ninguno de los tres benchmarks.

A juicio del autor y en base a los resultados presentados, consideramos que Google App Engine es la mejor alternativa de cómputo en la nube, dados sus mínimos tiempos de ejecución, sus mejores "tiempos máximos" y sus mejores tiempos promedios, además de su facilidad de desarrollo y despliegue gracias al SDK y plugin para el entorno de desarrollo Eclipse.

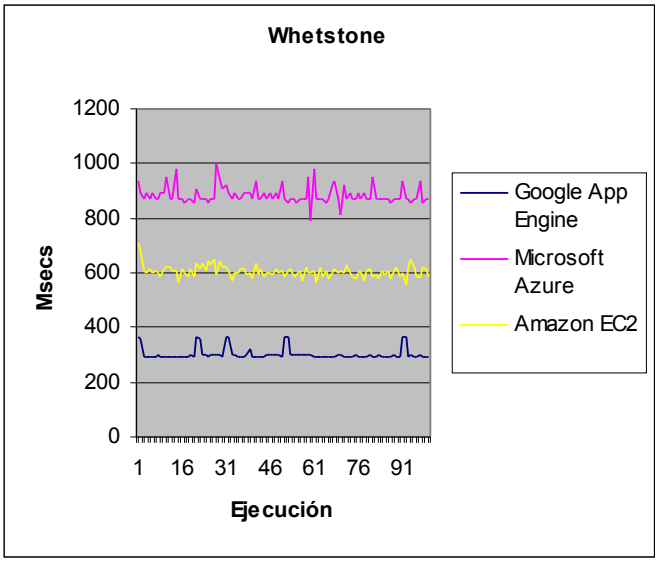

Fig. 2. Grafico comparativo Whetstone

En la figura 2 podemos evaluar sin lugar a dudas que App Engine realizó el mejor desempeño entre sus dos competidores (Azure y Amazon), sacando una amplia ventaja en los tiempos de ejecución.

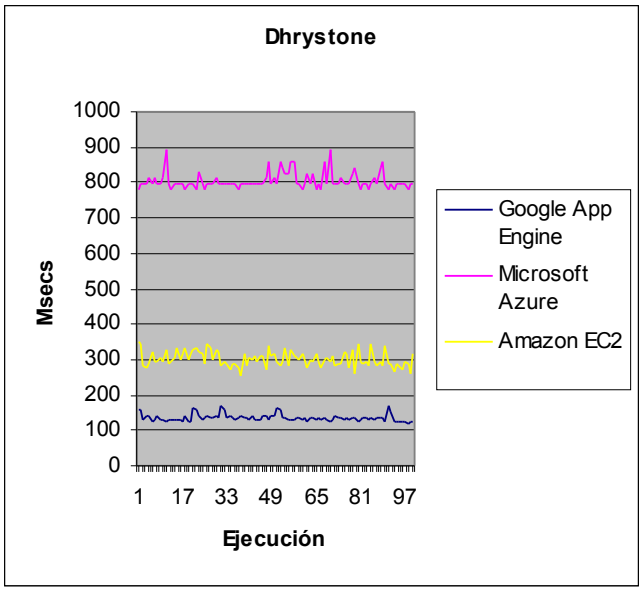

Fig. 3. Gráfico comparativo Dhrystone

En el gráfico de la figura 3 se aprecia claramente que Google App Engine supera sin problemas a Amazon EC2 y Microsoft Azure, con una amplia diferencia de tiempos a su favor.

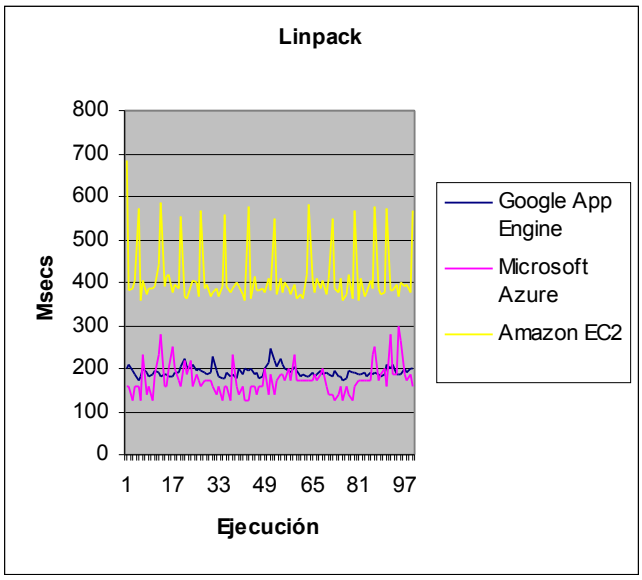

Fig. 4. Gráfico comparativo Linpack

En el gráfico anterior (fig. 4) se aprecian los picos cíclicos que sufre Amazon EC2 al ejecutar la prueba Linpack. En comparación a Azure y Amazon, el rendimiento de Google App Engine es más estable y predecible, ya que no sufre tantos picos ni sobresaltos.

En este caso, se podría decir que hay un empate entre Azure y App Engine, aunque la estabilidad promedio sugiere que Google se comportaría mejor a largo plazo.

6. Conclusion

En el transcurso del presente trabajo se lograron alcanzar los tres objetivos planteados al inicio de la investigación.

En primer lugar, se investigaron diferentes benchmarks existentes, buscando definir un conjunto de pruebas que permitiera obtener resultados útiles el problema particular del análisis de rendimiento para soluciones de Cloud Computing. A partir de las diferentes alternativas estudiadas, se optó por definir un conjunto de tres pruebas ampliamente difundidas (Whetstone, Dhrystone y Linpack) que sirvieran para analizar diferentes aspectos del rendimiento de un sistema informático.

Como segundo objetivo específico, a partir de los benchmarks elegidos por el autor, se desarrolló un Framework de pruebas que simplificara el desarrollo e implementación de las pruebas de software planteadas, incluyendo las interfaces con el usuario, la contabilización de los tiempos de ejecución y la presentación de resultados. La aplicación de software así obtenida cumplió la generación de resultados estandarizados en cada plataforma de Cloud Computing para los tres benchmarks planteados.

Consideramos además que el conjunto de benchmarks exhibido en el presente trabajo es útil como puntapié inicial para la evaluación y comparación de poder computacional en soluciones de Cloud Computing, ya que nos otorga la posibilidad de obtener mediciones objetivas del rendimiento de procesamiento computacional de la Nube en cuestión, al margen de lo que pueda informar la empresa sobre su propio producto.

Este Framework y su conjunto de benchmarks asociado se podrán aplicar a cualquier solución de Cloud Computing que permita la ejecución de un "Web Service" provisto por el usuario, donde sea necesario considerar cuestiones de performance de procesamiento computacional.

En cambio, este benchmark no debería aplicarse en la toma de decisiones para la selección de un proveedor de Cloud Computing si el núcleo de la aplicación que se ejecutará en la Nube no tiene un componente crítico de procesamiento computacional.

Con respecto a los aspectos económicos, este trabajo demuestra que la solución más costosa en el ámbito monetario no asegura la mejor performance a nivel de procesamiento para una aplicación en particular, por lo cual será importante evaluar con detenimiento que tipo de aplicación deseamos ejecutar en la Nube antes de elegir un proveedor tecnológico. Además del ahorro económico, realizando una evaluación adecuada de lo que deseamos ejecutar podremos incluso obtener mejores tiempos de respuesta y una optima performance frente a la competencia.

El último objetivo se alcanzó durante la evaluación y el análisis de los resultados generados por la aplicación de software codificada. A partir de la implementación de la aplicación de software desarrollada por el autor, se realizó la evaluación de los resultados obtenidos y se pudieron comprobar las diferencias de rendimiento computacional de cada una de las tres soluciones comerciales de Cloud Computing.

Como conclusión final, se destaca que el sistema propuesto demostró su utilidad concreta al encontrar diferencias significativas en la ejecución de benchmarks en soluciones reales de Cloud Computing.

7. Futuras Líneas de Investigación

Durante el desarrollo del presente trabajo se dejaron áreas relacionadas con Cloud Computing sin cubrir, que consideramos muy importantes de incluir en un futuro trabajo para dar un verdadero análisis holístico esta nueva tecnología. Con especial énfasis se debe profundizar en el desarrollo de pruebas objetivas para comparar las soluciones de acceso y persistencia de bases de datos dentro de la Nube, ya que toda aplicación empresarial actual interactúa casi indefectiblemente con grandes volúmenes de datos almacenados en tablas relacionales. La evaluación del acceso a datos, junto con la valoración del procesamiento computacional, otorgaría una vista ampliada desde donde se podría comparar con mayor objetividad todas las soluciones de Cloud Computing existentes.

Como dificultad asociada a este análisis, consideramos que el desarrollo de métricas homogéneas capaces de balancear y estandarizar los resultados de una forma objetiva para todas las arquitecturas y tecnologías disponibles, aportaría un gran valor agregado a las pruebas presentes y futuras sobre Cloud Computing.

A partir de la investigación desarrollada en el presente trabajo, el autor plantea las siguientes líneas de investigación futuras:

- La extensión del Framework para soportar benchmarks y pruebas relacionadas con el almacenamiento de datos en la nube.
- La optimización y e incorporación de otros benchmarks reconocidos al lote de pruebas para obtener resultados mas detallados de cada una de las soluciones de Cloud Computing.
- El desarrollo de pruebas y benchmarks inéditos que utilicen llamadas a funciones específicas de las APIs de los SDK disponibles para cada nube computacional.
- La implementación del Framework en otros lenguajes de programación como Python y PHP, que permitan comparar el rendimiento del conjunto de pruebas equivalente en una misma nube, pero en un lenguaje e intérprete diferente.

Referencias

[1] Amazon Web Services. (2009a). Advantages of Amazon EC2. Recuperado el 5 de septiembre de 2009 de http://aws.amazon.com/ec2/

[2] Amazon Web Services. (2009b). Amazon Elastic Compute Cloud: User Guide. Recuperado el 5 de septiembre de 2009 de http://aws.amazon.com/documentation/

[3] Amazon Web Services. (2009c). Amazon Simple Storage Service. Recuperado el 5 de septiembre de 2009 de http://aws.amazon.com/s3/
[4] Amazon Web Services. (2009d). AWS Toolkit for Eclipse. Recuperado el 5 de septiembre de 2009 de http://aws.amazon.com/eclipse/
[5] Armbrust M., Fox A., Griffith R., Joseph A., Katz R., Konwinski A. et al. (2009). Above the Clouds: A Berkeley View of Cloud Computing. Recuperado el 18 de agosto de 2009 de http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html
[6] Cearley D. & Feiman J., (2009). Economics of the Cloud: Business Value Assessments. Recuperado el 29 de septiembre de 2009 de http://www.gartner.com/DisplayDocument?id=1189413
[7] "Cloud Computing". (2009). Wikipedia. Recuperado el 23 de agosto de 2009 de http://en.wikipedia.org/wiki/Cloud_computing
[8] Chappell D., (2008). A Short Introduction to Cloud Platforms. Recuperado el 23 de agosto de 2009 de http://www.davidchappell.com/CloudPlatforms--Chappell.pdf
[9] Chappell D., (2009a). Introducing the Windows Azure Platform. Recuperado el 28 de agosto de 2009 de http://www.microsoft.com/windowsazure/
[10] Chappell D., (2009b). Introducing Windows Azure. Recuperado el 28 de agosto de 2009 de http://www.microsoft.com/windowsazure/
[11] Curnow H. & Wichmann B., (1976). A synthetic benchmark. Computer Journal. 19 (1), 43-49.
[12] "Dhrystone". (2009). Wikipedia. Recuperado el 26 de agosto de 2009 de http://en.wikipedia.org/wiki/Dhrystone
[13] Dongarra J. & Stewart G., (1984). Sources and Development of Mathematical Software. En Cowell W., (Ed.). LINPACK, A Package For Solving Linear Systems. (p. 20–48). New York: Ediciones Prentice-Hall.
[14] Google Inc. (2009a). App Engine Java Overview. Recuperado el 6 de septiembre de 2009 de http://code.google.com/intl/en/appengine/docs/java/overview.html
[15] Google Inc. (2009b). Google App Engine: Quotas. Recuperado el 6 de septiembre de 2009 de http://code.google.com/intl/en/appengine/docs/quotas.html
[16] Google Inc. (2009c). Google App Engine: Developer's Guide. Recuperado el 6 de septiembre de 2009 de http://code.google.com/intl/en/appengine/docs/
[17] Hayes B., (2008). Cloud Computing. Communications of the ACM. 51 (7), 9-11.
[18] "Linpack". (2009). Wikipedia. Recuperado el 26 de agosto de 2009 de http://en.wikipedia.org/wiki/Linpack
[19] Mäenpää J., (2009). Cloud computing with the Azure platform. Recuperado el 3 de septiembre de 2009 de http://www.cse.tkk.fi/en/publications/B/5/papers/Maenpaa_final.pdf
[20] Microsoft Corporation. (2009). Windows Azure SDK. Recuperado el 28 de agosto de 2009 de http://msdn.microsoft.com/es-ar/library/dd179367(en-us).aspx
[21] Sun Microsystems, Inc. (2009). Introduction to Cloud Computing Architecture. Recuperado el 22 de agosto de 2009 de http://www.sun.com/featured-articles/CloudComputing.pdf
[22] Weiss A., (2002). Dhrystone Benchmark. History, Analysis, Scores and Recommendations. Recuperado el 26 de agosto de 2009 de http://www.johnloomis.org/NiosII/dhrystone/ECLDhrystoneWhitePaper.pdf
[23] "Whetstone (benchmark)". (2009). Wikipedia. Recuperado el 26 de agosto de 2009 de http://en.wikipedia.org/wiki/Whetstone_(benchmark)