Rendimiento de Aplicaciones Web en plataformas de contenedores de código abierto

Lic. Gabriel Antonio Coronel, Mgter Oscar Adolfo Vallejos
Universidad de la Cuenca del Plata
email: gabriel.c190@gmail.com; oval_lejos@hotmail.com

Abstract

Cloud computing makes extensive use of virtual machines, because they allow the workload is isolated and also, the use of resources is somewhat controlled. Currently, the additional levels of abstraction involved in virtualization allow to reduce workload performance. The most recent advances in container-based virtualization simplify the deployment of software applications without this implying the loss of control over the resources assigned to different applications. In this paper, we analyze the performance of a web application running in the containers of the Docker platform, as well as that of another free software platform, KVM, and a host operating system.

Resumen

La computación en la nube hace un uso extensivo de máquinas virtuales, pues permiten que la carga de trabajo esté aislada y además, el uso de los recursos este algo controlado. En la actualidad, los niveles adicionales de abstracción involucrados en virtualización permiten reducir el rendimiento de carga de trabajo. Los avances más recientes en la virtualización basada en contenedores simplifican la implementación de aplicaciones de software sin que esto implique la perdida el control sobre los recursos asignados a las diferentes aplicaciones. En este documento, se analiza el rendimiento de una aplicación web funcionando en los contenedores de la plataforma Docker[1], como también el de otra plataforma de software libre, KVM, y un sistema operativo anfitrión.

Palabras claves: máquinas virtuales, virtualización basada en contenedores, contenedores de código abierto.

Introducción

En la actualidad, en el mundo de las tecnologías de información, hay ciertas problemáticas que afectan a diversas empresas y organizaciones en cuanto a la tecnología utilizada para su plataforma de computación en la nube. Además de poseer el equipamiento para ofrecer procesamiento y almacenamiento suficiente para cubrir los requerimientos del negocio, son necesarios los medios para otorgar recursos de forma eficaz y eficiente cuando el cliente los requiera. Con la virtualización los servidores estarán preparados para responder a cambios en las cargas de trabajo y al tener cada aplicación aislada con su propio servidor virtual se puede evitar que una aplicación impacte otras aplicaciones al momento de realizar mejoras o cambios.

Aún así, es posible encontrar ciertas deficiencias a la hora de implementar virtualización en una infraestructura. Entre las mismas se encuentran la sobrecarga de ejecutar múltiples máquinas virtuales, la dificultad de la administración de dependencias de las aplicaciones, problemas no presentes en entornos de desarrollo de software pero presentes en entornos de producción.

Estos problemas pueden solucionarse mediante el empleo de los contenedores de aplicaciones, que utilizan una virtualización de sistema operativo, pero surge la problemática del desempeño que pueda acarrear esta tecnología, de forma que existe la posibilidad de convertirse en una mala decisión implementar esta plataforma.

Docker es una forma de encerrar los servicios en entornos aislados, llamados contenedores, de modo que puedan ser empaquetados con todo lo que necesitan en términos de las bibliotecas y dependencias y el desarrollador puede estar seguro de que el servicio se ejecutará donde sea que Docker funcione [1]. Docker trae varias consideraciones que en las tecnologías anteriores no se advierten. Entre ellas, es que los contenedores son más fáciles de implementar y utilizar que los métodos anteriores, más allá de la ventaja de la asociación de Docker con otras empresas de contenedores, tales como Canonical, Google, Red Hat y Parallels, sobre su componente clave libcontainer de código abierto, que ha traído la estandarización, algo muy necesario en los contenedores.

Considerando que se trata de una tecnología nueva en constante cambio, no hay un conjunto

de buenas prácticas establecidas para que las empresas puedan migrar fácilmente a esta tecnología en estos días, sobre todo si no disponen de recursos como las empresas anteriormente mencionadas. Otra cuestión reside en la falta de documentación de rigor científico para sostener las ventajas de rendimiento que brindan los contenedores libres funcionando con la plataforma Docker.
Teniendo en cuenta lo anterior se realizó este trabajo en el que se llega a una conclusión de manera empírica acerca de las ventajas de performance de Docker comparado con una máquina virtual de QEMU-KVM y un sistema operativo anfitrión. KVM simplemente convierte el kernel de Linux en un hipervisor cuando se instala el módulo kernel de KVM…Para las emulaciones de entrada y salida, KVM utiliza un software de usuario, QEMU; este es un programa que hace emulación de hardware[2]. En este trabajo se seleccionaron una serie de métricas que se tuvieron en cuenta para realizar pruebas de rendimiento de una aplicación web funcionando con las tecnologías de virtualización mencionadas anteriormente, utilizando el software para medir el rendimiento de una aplicación web (rails perf test) y la herramienta de monitoreo de sistema de Ubuntu Linux, que proporciona datos precisos del sistema operativo anfitrión.

Método

Hoy en día, en un mundo cada vez más competitivo, el manejo de información se ha convertido en un factor determinante para el negocio de las empresas. Una buena elección de las tecnologías para su plataforma de sistemas de IT (tecnologías de información) posibilitará que el negocio tenga más posibilidades de asegurar una posición exitosa en el mercado.

En este contexto, las empresas inmersas en la computación en la nube, que se dedican a implementar software como servicio, dependen del rendimiento de su aplicación web para que su negocio prospere.

Si no se mide, el juicio puede basarse solamente en la evaluación subjetiva. Para permanecer en el mercado las empresas necesitan datos del desempeño de su servicio para poder hacer evaluaciones objetivas, por lo tanto, resulta muy importante la medición del software. El *IEEE Standard Glosary of Software Engineering Terminology* [3] define métrica como "una medida cuantitativa del grado en el que un sistema, componente o proceso posee un atributo determinado". Las métricas de software proporcionan a los gerentes de desarrollo de software y administradores de proyectos una visión sensata de lo bien que sus aplicaciones funcionan en realidad. Aisladas, las métricas son sólo representaciones gráficas o visuales de datos. Sin embargo, mediante el análisis de las métricas, los desarrolladores de software pueden descubrir patrones, tendencias y problemas que afectan al rendimiento de las aplicaciones.

En los últimos años muchas empresas han implementado la virtualización de software para aportar mayor rendimiento a su aplicaciones. La virtualización permite que una sola computadora contenga varias máquinas virtuales, cada una de las cuales puede llegar a ejecutar un sistema operativo distinto [4]. Siguiendo este camino aparece la plataforma de contenedores de Docker que utiliza una virtualización con menor sobrecarga, a diferencia de las otras máquinas virtuales.

Con esto, un equipo de desarrolladores puede llevar a cabo una aplicación como si utilizaran la misma máquina, logrando un mejor proceso para manejar las dependencias y facilitando el traslado del software al servidor u otras máquinas.

Una migración a contenedores podría facilitar el proceso de despliegue de una aplicación web. Con lo cual se reduciría el tiempo que le cuesta a una organización realizar el despliegue de su software. Al mismo tiempo resultaría más simple el trabajo de mantener entornos de desarrollo con diferentes sistemas operativos de forma estable. Según el servicio de monitoreo Datadog[5] en las empresas que adoptan Docker, los contenedores tienen una vida útil media de 2,5 días, mientras que en todas las empresas, las máquinas virtuales tradicionales en la nube tienen una vida media de 23 días. Lo cual indica la simplicidad para subir una nueva instancia de una aplicación en un contenedor y reemplazar otra comparado con las VM.

Este trabajo provee una comparación actualizada de una aplicación web con el uso de contenedores, máquina virtual y sistema operativo huésped utilizando hardware reciente y software con una serie de pruebas para analizar sus respectivos rendimientos.

Como objetivo general se evaluo las ventajas y desventajas con métricas de rendimiento para una aplicación web utilizando un sistema operativo huésped, un sistema operativo virtualizado y un contenedor.
Para ello fue necesario: a) Desarrollar una aplicación web que funcione en la plataforma de contenedores Docker, en un sistema operativo virtualizado y sistema operativo huésped; b) Seleccionar, aplicar y analizar las métricas de rendimiento comparando el uso sobre el sistema operativo, un sistema operativo virtualizado y un contenedor; c) Analizar las ventajas y desventajas relacionadas con el manejo de dependencias.

Se realizaron las siguientes etapas:

- Se realizó una revisión sobre las tecnologías para poder desarrollar una aplicación web, es decir, los frameworks web que se utilizan actualmente en el desarrollo de software ágil, teniendo como resultado la definición de qué tecnologías utilizar a la hora de llevar a cabo el desarrollo.
- Se llevó a cabo el desarrollo del software funcionando con los contenedores de Docker. Este software luego será ejecutado también en una máquina virtual y un sistema operativo huésped.
- Se realizo una búsqueda de las métricas de software enfocadas en el rendimiento y se seleccionaron las más adecuadas.
- Posteriormente, se aplicaron las métricas para contrastar entre estos diferentes entornos, con lo cual se obtuvieron los resultados, que fueron analizados.
- Finalmente, se elaboraron las conclusiones correspondientes.

Desarrollo de la experiencia

Se seleccionaron los diez primeros frameworks web más utilizados en la actualidad, considerando aquellos que incluyan la programación del lado del servidor de acuerdo a la web hotframework[6].
De esta selección, se opto la utilización del framework Ruby on Rails versión 4.2, debido a su interfaz amigable y a la concreción de un prototipo en un periodo corto de tiempo, con lo cual resulta muy productivo.

La aplicación web consiste en una librería en línea, es decir, permite comprar libros por internet de una librería para luego ser enviados al cliente. Dicha aplicación sirve como un caso para poder ejecutar aplicación usando un sistema operativo anfitrión, un contenedor de Docker y una máquina virtual para poder obtener las métricas relacionadas al rendimiento. El desarrollo fue llevado a cabo utilizando una metodología hibrida de extreme programming. Como lenguaje se utilizo una implementación de Ruby en Java, lo que permite una mejor concurrencia con aplicaciones exigentes.
Docker es multiplataforma y para poder ejecutarse en Windows y Mac OS debe utilizarse un hipervisor que ejecute una máquina virtual de Linux en el cual estará en funcionamiento del contenedor, por lo tanto la manera más eficiente para esto es utilizar un sistema operativo Linux. Se seleccionó para esto el sistema Ubuntu por ser un sistema amigable y con mucha documentación al respecto.

Durante la ingeniería de requerimiento se utilizaron el diagrama de flujo de datos, diagramas de casos de uso y diagrama de clases, que fueron refinados en la etapa del diseño.
Después de que se hayan hecho los diagramas, no se inició directamente la codificación, sino que se desarrolló una serie de pruebas unitarias y una prueba de integración, para cumplir con esa prueba se llevo a cabo un desarrollo iterativo e incremental. Para estas pruebas se utilizó la herramienta que viene con el framework, llamada MiniTest, la cual es una mejora de la herramienta original de testing de Rails Test::Unit. Se eligió este software al ser el recomendado por quienes intervienen en el desarrollo del framework Ruby on Rails y tener una sintaxis sencilla para escribir los tests. Una vez que el código estuvo terminado, se le aplicó de inmediato una prueba unitaria, con lo que se obtiene retroalimentación instantánea.

Para poder utilizar un contenedor y ejecutar la aplicación web en un dicho es necesario realizar un Dockerfile que contenga las instrucciones necesarias para crear una imagen de sistema operativo y posteriormente un contenedor (ver **Figura 1)**. Para poder generar una imagen realmente pequeña para ejecutar la aplicación web, se utilizó la versión 17 de Docker (Edición Comunitaria) y 3.6 de Alpine Linux. Esta distribución incluye sólo los archivos mínimos necesarios para arrancar y ejecutar el sistema operativo.

Posterior a eso se incluye la configuración para instalar Ruby on Rails y las librerías necesarias. El último paso consiste en ejecutar la instrucción que construya la imagen y ejecuta el contenedor.

```
FROM jruby:9.1.7-alpine

MAINTAINER Myself <gabriel@gmail.com>

ENV BUILD_PACKAGES="curl-dev ruby-dev build-base" \
    DEV_PACKAGES="zlib-dev libxml2-dev libxslt-dev tzdata yaml-dev sqlite-dev " \
    RUBY_PACKAGES="ruby ruby-io-console ruby-json yaml nodejs" \
    RAILS_VERSION="4.2.5"

RUN \
  apk --update --upgrade add $BUILD_PACKAGES $RUBY_PACKAGES $DEV_PACKAGES && \
  gem install -N bundler

RUN gem install -N nokogiri -- --use-system-libraries && \
  gem install -N rails --version "$RAILS_VERSION" && \
  echo 'gem: --no-document' >> ~/.gemrc && \
  cp ~/.gemrc /etc/gemrc && \
  chmod uog+r /etc/gemrc && \

  # cleanup and settings
  bundle config --global build.nokogiri  "--use-system-libraries" && \
  bundle config --global build.nokogumbo "--use-system-libraries" && \
  find / -type f -iname \*.apk-new -delete && \
  rm -rf /var/cache/apk/* && \
  rm -rf /usr/lib/lib/ruby/gems/*/cache/* && \
  rm -rf ~/.gem


RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app
COPY . /usr/src/app
RUN bundle install
EXPOSE 3000

# The main command to run when the container starts. Also
# tell the Rails dev server to bind to all interfaces by
# default.
CMD ["bundle", "exec", "rails", "server", "-b", "0.0.0.0"]
```

Figura 1: Dockerfile utilizado para la aplicación

A continuación se seleccionaron métricas para la medición de las siguientes características:

- Medir el tiempo del mundo real transcurrido durante la prueba de funcionamiento (wall time). Diferencia entre el momento en el que termina una tarea y la hora a la que comenzó la tarea. Se ve afectada por cualquier otro proceso al mismo tiempo que se ejecutan en el sistema.
- Medir el tiempo del usuario. Cantidad de tiempo que la CPU gastó en modo de usuario, es decir, dentro del proceso. Esto no se ve afectado por otros procesos y por el tiempo que posiblemente se gasta bloqueado.
- Medir la cantidad de memoria utilizada para el caso de prueba de rendimiento. El uso de la memoria proporciona la información sobre la cantidad de espacio libre en disco.
- Medir la cantidad de tiempo gastado en recolección de basura para el caso de prueba de rendimiento.

Se realizaron varias pruebas diferentes, las cifras que se muestran son las mejores que se obtuvieron. Esto se pudo realizar mediante la herramienta rails perf test versión 0.0.7. Esta herramienta permite hacer un tipo especial de pruebas de integración, diseñada para probar el rendimiento de una aplicación web hecha con el framework web Ruby on Rails. Se eligió este software al ser un proyecto con desarrollo activo y open source. Con esto se realizó una prueba de estrés para exigir al sistema los más posible. La prueba es similar a un test de integración, se carga la página inicial pero con una gran cantidad de productos y se realiza una compra. Cuando se utilizó las herramientas de virtualización se tuvo en cuenta el estado del sistema operativo anfitrión(consumo de CPU y memoria RAM) con la herramienta de monitoreo del sistema, la cual viene por defecto en el sistema operativo Ubuntu.

Resultados Obtenidos

Para el sistema operativo anfitrión se utilizo el sistema operativo Ubuntu 16.04 con el kernel 4.10, que luego seria el anfitrión de la máquina virtual y el contenedor, se realizó la prueba de rendimiento (ver **Tabla 1**).

Wall time	Tiempo de usuario	Memoria
124 mili segundos	57 mili segundos	7,1 megabytes

Tabla 1: Resultados sistema operativo

Para el sistema operativo invitado se utilizo una máquina virtual de Ubuntu creada con QEMU-KVM en su versión 2.5, administrando dicha máquina virtual con el programa virt-manager versión 1.3.2 e instalando la aplicación web en dicha máquina se realizó la prueba de rendimiento (ver **Tabla 2**). Utilizando el monitor de sistema de Ubuntu se aprecia que los CPU llegaron a un pico de 100%. El consumo de memoria de sistema operativo tuvo un máximo de 2,1 gigabytes y se obtuvieron la métricas correspondientes. En estado de reposo la máquina virtual consume 1,9 gb (gigabytes).

Wall time	Tiempo de usuario	Memoria
369 mili segundos	137 mili segundos	6,8 megabytes

Tabla 2: Resultados de máquina virtual

Midiendo el sistema sin exigirlo después de haberlo iniciado, se obtuvo estas cifras: 33% de uso de CPU y 1,9 gigabytes de memoria ram. Al momento de exigirlo llego a un 100% de uso de CPU con 2,1 gigabytes de memoria.
Utilizando un contenedor Docker a partir de una imagen de sistema operativo personalizada para las dependencias de la aplicación web se realizó la prueba de rendimiento (ver **Tabla 3**). En este caso los CPU llegaron a un pico de 95% y 97%. El consumo de memoria de sistema operativo supero ligeramente al del hipervisor y se obtuvieron la

métricas correspondientes. En estado de reposo el contenedor consume 697,8 mb (megabytes).

Wall time	Tiempo de usuario	Memoria
133 mili segundos	67 mili segundos	6,3 megabytes

Tabla 3: Resultados de Docker

Los resultados de la prueba son casi iguales que en el sistema operativo anfitrión.
Midiendo el sistema sin exigirlo después de haberlo iniciado, se obtuvo estas cifras: 0,13% de uso de CPU y 697,8 megabytes de memoria ram. Al momento de exigirlo llego a un 97% de uso de CPU con 2,2 gigabytes de memoria. En ese momento de exigencia el consumo de memoria del sistema operativo fue un poco mayor que en el caso de la máquina virtual.

Conclusión

Al finalizar este trabajo, se puede concluir que mientras QEMU-KVM es una solución implementada con mayor tiempo en desarrollo, teniendo en cuanto los resultados obtenidos con la herramienta de pruebas de rendimiento rails perf test, Docker es capaz de proveer un mejor rendimiento utilizando una menor cantidad de recursos, siendo la performance más cercana al de la aplicación funcionando sin virtualización, si bien al momento de ser exigidas ambas plataformas, con Docker el sistema operativo mostró una leve mayoría en uso de memoria. También se debe tener en cuenta que si bien hay alternativas comerciales de hipervisores, en este caso QEMU-KVM es exclusivo para un sistema operativo Linux mientas que Docker cambió sus componentes para poder lograr contenedores en otras plataformas.
En el apartado de la administración de las dependencias para ejecutar la aplicación web, Docker tiene una manera más eficiente al tener el proceso automatizado declarando las dependencias necesarias en el Dockerfile mientras que en el caso de QUEMU-KVM es un proceso manual por parte del usuario.
Llevar a cabo este proyecto, entendemos, contribuye disponer de un sostén científico respecto del rendimiento de contenedores de código abierto utilizados para virtualizar aplicaciones web. Con lo cual sería un factor importante en el momento que una organización decida migrar a esta tecnología.

Referencias Bibliográficas

[1] Hane, O. (2015). *Build Your Own PaaS* with Docker. Birmingham, Reino Unido. Packt Publishing.
[2] Mukhedkar P. (2016). *Mastering KVM Virtualizatio*n. Birmingham, Reino Unido. Packt Publishing.
[3] *IEEE Standard Glossary of Software Engineering Terminology.* (1993). IEEE.
[4] Tanenbaum, A.(2009). *Sistemas Operativos Modernos*. Naucalpan de Juárez, México. Pearson Education.
[5] (2017). 8 Surprising facts about real Docker adoption.
[6] (2017). Hot Frameworks. Recuperado de: https://hotframeworks.com/