Ingeniería de Servidores (2018-2019)
Grado en Ingeniería Informática
Universidad de Granada

Práctica 4 : Benchmarking y Ajuste del Sistema

18 de noviembre de 2018

Índice

OBJETIVOS MÍNIMOS

1. Conocer varios benchmarks para diferentes servicios.
2. Saber comparar distintas configuraciones (o implementaciones) de servicios en base a benchmarks.
3. Aplicar un test de carga a una aplicación basada en microservicios.
4. Conocer y saber cómo modificar el valor algunos parámetros que pueden mejorar las prestaciones.

Contenido de las lecciones

1. Benchmarks suites y Jmeter

1. Introducción

Como ejercicio final tras haber instalado, configurado y monitorizado servicios, vamos a aplicar una carga que nos pueda servir cómo indicador para ver cómo podría responder nuestro sistema. En base a los resultados obtenidos (y monitorizados) consideraremos también algunos parámetros que puedan mejorar las prestaciones ante cargas concretas. En primer lugar estudiaremos un conjunto de benchmarks para luego centrarnos en uno concreto para un servicio que ya conocemos y que trataremos de optimizar su comportamiento.

2. Benchmarks

Hay muchos para cada tipo de servicio (p.ej. Para DNSs: NameBench y GRC's DNS Benchmark, `http://sourceforge.net/directory/os:linux/?q=benchmark`) pero puede resultar más interesante programar uno para analizar algún parámetro concreto que deseemos. Siempre debemos tener en cuenta un mínimo de cuestiones al hacerlo:

1. Objetivo del benchmark.
2. Métricas (unidades, variables, puntuaciones, etc.).
3. Instrucciones para su uso.
4. Ejemplo de uso analizando los resultados.

2.1. Phoronix

Phoronix es una plataforma que permite ejecutar un conjunto de benchmarks bajo la agrupación openbenchmarking.org. La aplicación puede instalarse a través de los gestores de paquetes ya vistos en los guiones anteriores.
Una vez que conoce los benchmarks, puede pasar a seleccionar algunos de ellos e instalarlos (no todos los listados son los que están instalados).
Otra funcionalidad interesante es la interfaz phoromatic que permite orquestrar y automatizar la ejecución de benchmarks en múltiples máquinas.
Podemos hacer uso de una interfaz web mediante la instalación de un navegador, p.ej. firefox, y ejecutando el comando correspondiente indicado en la documentación [6]. La interfaz realiza una monitorización de la ejecución que podemos contrastar simultaneamente con la que se realiza mediante Zabbix 2.1.

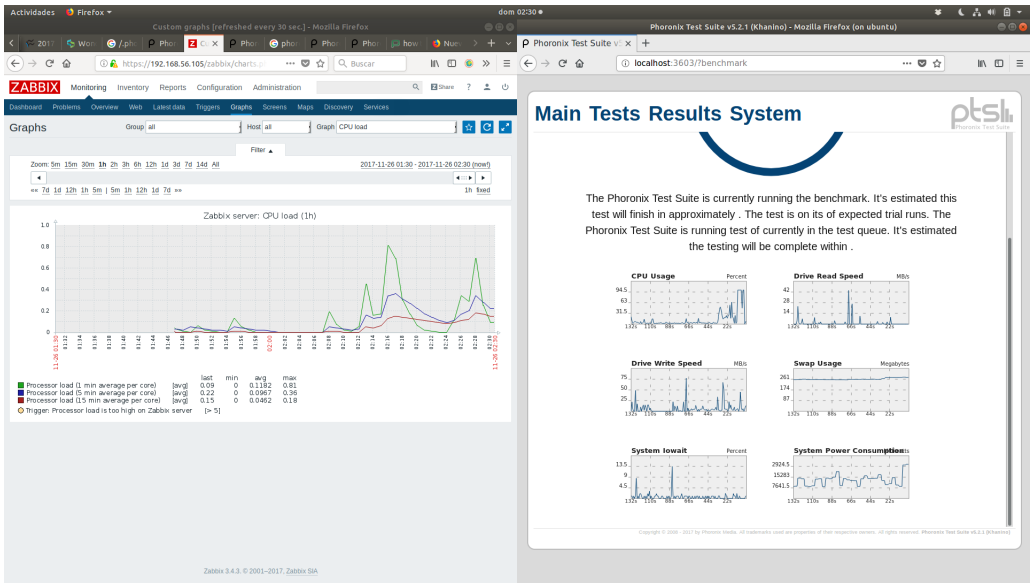

Figura 2.1: Ejemplo de monitorización con Zabbix y phoronix durante la ejecución de un benchmark

2.2. ab y Jmeter

Dentro de los benchmarks más populares para servidores web podemos encontrar la herramienta para "Apache HTTP server benchmarking "(comando ab) [1]. Concretamente, su misión es mostrar cuantas peticiones por segundo el servidor de HTTP (puede ser Apache, Nginx, IIS o cualquier otro) es capaz de servir.
Dentro de los parámetros básicos que usted debe conocer encontramos la opción que especifica la concurrencia así como el número de peticiones. Usted debe probar a ejecutar el benchmark (monitorizar su ejecución en el cliente y en el servidor) con sus máquinas virtuales (Ubuntu y CentOS) y obtener conclusiones respecto al tipo de concurrencia que es capaz de generar ab además de comparar resultados.

Aunque tenemos muchas posibilidades usando ab de manera correcta, hay otras herramientas que nos permiten hacer tests más complejos, concretamente vamos a ver Jmeter `http://jmeter.apache.org/` [3]. Este software se autodefine como una aplicación "*...designed to load test functional behavior and measure performance...*", es decir, no establecen ningún tecnicismo concreto ya que pueden medir y generar carga de forma genérica para varios elementos.
Ha sido comparado por la empresa Octoperf con Gatling (otra herramienta popular para realizar test de estrés [5]) obteniendo la conclusión de que ambos tienen un comportamiento y capacidades similares [8].
Algunas características interesantes de Jmeter es que puede crear concurrencia real debido a la posibilidad de usar varias hebras dentro de la mismas CPU así como distribuir la creación de carga en varias máquinas. La posibilidad de ejecutarse en modo de línea de comandos permite aligerar la carga de la máquina que está generando las peticiones al servidor además de permitir la automatización de ciertos tests. También es muy interesante la funcionalidad que nos permite configurar como proxy de nuestro navegador a Jmeter para que vaya registrando nuestra navegación como usuarios y luego podamos replicar esas peticiones de manera automática y paralela.
Tras probar un test básico para una web [4], utilizaremos Jmeter para hacer un test sobre una aplicación que ejecuta sobre dos contenedores (uno para la BD y otro para la aplicación en sí). El código está disponible en `https://github.com/davidPalomar-ugr/iseP4JMeter` donde se dan detalles sobre cómo ejecutar la aplicación en una de nuestras máquinas virtuales.
En lo que se refiere al futuro, puede ser interesante seguir el proyecto de código abierto Locust `https://locust.io/` que permite escribir en código (con Python) los tests sin necesidad de interfaces permitiendo comprobar la escalabilidad sin límites en lo que se refiere a la creación de hebras.

3. Ajuste del sistema y de servicios

La modificación de los parámetros del sistema es una tarea compleja pues existen una gran cantidad de éstos para otra gran cantidad de subsistemas que, además, están relacionados. Tan solo en lo referente a los parámetros del kernel, tenemos un gran abanico de variables cuyos valores pueden ser números naturales, siendo muy difícil establecer o estimar una aproximación al óptimo (teniendo en cuenta que éste puede variar dependiendo de la carga). Algunos de estos elementos ajustables del kernel pueden ser accedidos y modificados a través del sistema de archivos (/proc/sys). Modificando los valores dentro de los archivos, es posible cambiar el comportamiento del sistema sin tener que reiniciarlo. Dentro de /proc/sys encontramos una estructura de directorios que separa los archivos de los distintos subsistemas (fs: sistema de archivos, kernel: kernel, vm: memoria virtual, dev: dispositivos, etc.)
Para evitar errores o valores inválidos, es recomendable usar el comando sysctl para modificar los parámetros del kernel.

3.1. Optimizando un servidor web

Como ya estamos familiarizados con la pila LAMP, vamos a considerar algunos parámetros y configuraciones interesantes que nos permitan obtener un mayor rendimiento.
No solo hay que tener en cuenta los elementos del sistema operativo y de los servicios en ejecución sino que cada código tiene unas características que nos obligan a monitorizar durante un tiempo la actividad e ir ajustando los valores en base a los resultados. Por ejemplo, si queremos tener una plataforma para enseñanza como puede ser moodle, ellos mismos ya tienen algunas sugerencias que, a posteriori, podremos modificar, pero son un buen punto de partida [7] y que incluyen las recomendaciones de la documentaciónd de Apache [2]. Otro artículo en la misma línea es el incluido en la documentación de Wordpress, un Content Management System (CMS) de los más populares [9].
Con esta información usted podría modificar los parámetros de configuración de Apache, PHP o MariaDB para observar un cambio en el comportamiento del servidor (CentOS o Ubuntu server) mediante la aplicación de un benchmark y analizando el cambio en las prestaciones o mediante el análisis de datos de monitorización ante una carga aplicada. **Esta tarea es opcional**.

Referencias

[1] The Apache Software Foundation. ab - apache http server benchmarking tool. `https://httpd.apache.org/docs/2.4/programs/ab.html`, 2017. [Online; consultada 14-September-2017].

[2] The Apache Software Foundation. Apache performance tuning. `http://httpd.apache.org/docs/current/misc/perf-tuning.html`, 2017. [Online; consultada 14-September-2017].

[3] The Apache Software Foundation. Best practices. `http://jmeter.apache.org/usermanual/best-practices.html`, 2017. [Online; consultada 14-September-2017].

[4] The Apache Software Foundation. Building a web test plan. `http://jmeter.apache.org/usermanual/build-web-test-plan.html`, 2017. [Online; consultada 14-September-2017].

[5] Gatling. Gatling. `http://gatling-tool.org/`, 2017. [Online; consultada 14-September-2017].

[6] Phoronix Media. Phoronix test suite. `https://www.phoronix-test-suite.com/documentation/phoronix-test-suite.html`, 2017. [Online; consultada 14-September-2017].

[7] Moodle. Performance recommendations. `https://docs.moodle.org/33/en/Performance_recommendations`, 2017. [Online; consultada 14-September-2017].

[8] Jérôme Loisel CTO Octoperf. Jmeter vs gatling tool. https://octoperf.com/blog/2015/06/08/jmeter-vs-gatling/#gist23106684, 2015. [Online; consultada 14-September-2017].

[9] WordPress. Wordpress optimization. https://codex.wordpress.org/WordPress_Optimization, 2017. [Online; consultada 14-September-2017].