**https://www.overleaf.com/13966507ptwwfpmqrcdc**

**Introducción**

La siguiente práctica tiene como objetivo realizar las pruebas de rendimiento de un servidor web Apache previamente instalado en una máquina virtual que nos ha provisto la Universidad.

Las pruebas de rendimiento son aquellas técnicas que nos devolverán como resultado una apreciación aproximada del rendimiento del propio sistema bajo unas condiciones específicas de trabajo. En nuestro caso, evaluaremos los tiempos de respuesta y la productividad, que miden respectivamente el retraso dentro de la red para los paquetes individuales y la cantidad de trabajo útil por unidad de tiempo en el entorno de carga.

Dicha evaluación se realizará por medio de diferentes pruebas de carga. Las pruebas de carga se utilizan principalmente para comprobar el comportamiento de un sistema ante una cantidad de peticiones esperada. Nuestras pruebas variarán en función del número de peticiones y clientes concurrentes que se establezcan para cada escenario. En el Anexo 1 se detallan las pruebas realizadas.

Después de ejecutar las 5 pruebas para cada caso, se recogerán los datos pertinentes y se introducirán en unas tablas que permitan analizar y sacar conclusiones sobre la productividad y rendimiento del servidor web.

**Sistema en Evaluación**

El sistema a evaluar es una máquina de laboratorio asignada a nuestro grupo que tiene como S.O. *Ubuntu Server 16.04 LTS*. Accederemos a dicha máquina mediante ssh o el cliente gráfico X2GO para poder instalar las dependencias y establecer las configuraciones que sean necesarias.

Además, el software usado para el servidor será Apache, que se trata de un servidor web HTTP de código abierto, para plataformas UNIX, Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1. Este servidor es usado, entre otras cosas, para webs estáticas y dinámicas como las que implementan los casos de prueba de esta práctica.

Este software provee una funcionalidad llamada Apache Benchmark que permite lanzar las pruebas que sean necesarias para evaluar dicho servidor web, dicho comando será ejecutado de la siguiente manera:

$ab -n -c hostname[:port]

La opción -n indica el número de peticiones que se quieren lanzar y -c el número de clientes concurrentes que se establecen para dicha prueba. En cuanto al hostname, nosotros tendremos que sustituirlo por la dirección de nuestra máquina virtual, la cual es: virtual.lab.inf.uva.es con el puerto 31222.

Los resultados que devuelve Apache Benchmark se pueden observar en la figura 1, cuya salida corresponde a un test cualquiera sobre el servidor.

(INCLUIR IMAGEN RESULTAD AB)

Para apoyar los resultados que nos provee **ab** hemos utilizado una herramienta llamada ***SAR*** (System Activity Reporter), la cual se ejecuta del lado del servidor y con la que podremos obtener informes sobre el rendimiento de todo el sistema. El comando tiene la siguiente estructura:

$ sar [interval] -D

En primer lugar, podemos sustituir interval por un número que indicará el tiempo que se desea recibir información sobre el servidor. La opción -D sirve simplemente para poder guardar los resultados en un archivo independiente. Con todo ello, la salida de SAR será del siguiente estilo:

(INCLUIR IMAGEN RESULTAD sar)

**Resultados y análisis de los resultados.**

Realización de las Pruebas (Script Utilizado)

Para realizar dichas pruebas hemos decidido crear un script bash que aglutine todas las ejecuciones necesarias para cada uno de los tres casos de prueba y así poder lanzar todos los test a la vez.

También cabe destacar que el script será creado y lanzado desde una máquina del laboratorio personal de la Universidad (jair.lab.inf.uva.es), lo que reducirá el riesgo de posibles latencias a la hora de ejecutarlo en caso de que el servidor pueda rechazar alguna prueba o no pueda completar las peticiones.

El script ha sido lanzado 5 veces a distintas horas del día para obtener unos resultados sin posibles anomalías que falseen los datos obtenidos, consiguiendo así información más objetiva independiente de la hora y la posible carga de la red interna de la UVA.

Separaremos cada intensidad de carga con un sleep para que la ejecución sea lo más precisa posible. Dicho sleep será de 30 segundos. Este sleep tendrá que ser tenido en cuenta a la hora de analizar los resultados obtenidos en el SAR.

Las pruebas han sido realizadas a las siguientes horas:

Prueba 1 -> 12:54 h

Prueba 2 -> 18:45 h

Prueba 3 -> 00:27 h

Prueba 4 -> 10:08 h

Prueba 5 -> 14:45 h

Dicho script tendrá la siguiente forma:

(INCLUIR IMAGEN SCRIPT AB)

Previamente a la ejecución de los test, en la parte de servidor lanzamos el comando SAR, que recogerá cada 2 segundos información sobre el estado y rendimiento del servidor, terminando la ejecución justo después de terminar la prueba. Dicho script será de la siguiente manera:

$ sar 2 -D > /home/usuario/DatosSAR/

Resultados

De entre todos los resultados obtenidos en cada prueba hemos decidido destacar y quedarnos con dos de ellos: **Total connection times** que medirá el tiempo de respuesta de cada petición y **Time per request** que medirá la productividad del sistema (ver tablas en anexo).

A partir de ellos podremos analizar de manera conjunta cada uno de los valores necesarios para determinar el rendimiento y productividad del servidor en el que estamos realizando las pruebas.

En cuanto a los datos del sar tenemos que son los siguientes para cada prueba:

Prueba 1:

Prueba 2:

Prueba 3:

Prueba 4:

Prueba 5:

Análisis

A partir de los resultados arrojados en el apartado anterior, y basándonos en los valores que hemos decidido destacar, creemos conveniente la necesidad de desglosar y estudiar los tres aspectos más importantes que facilitarán la compresión del rendimiento del servidor web. Estos son tiempo de respuesta, productividad y SAR.

Tiempo de Respuesta

Analizar tiempo de respuesta en función de cada caso de prueba/intensidades/ensayo.

En función de la figura , observamos que los datos obtenidos sobre los tiempos de respuesta muestran una tendencia ascendente según crecen los niveles de concurrencia en los test establecidos para cada intensidad de carga.

(INTRODUCIR GRÁFICA COMPARATIVA DE MEDIAS)

Como podemos comprobar también, el número de peticiones hace variar muy levemente el tiempo de respuesta. Por el contrario, como hemos descrito anteriormente y basándonos en las tablas de los anexos , para el servidor si que tiene un verdadero impacto la subida en los clientes concurrentes haciendo que este disminuya de manera notoria su rendimiento frente a las pruebas ejecutadas..

Incluso comprobamos que en las intensidades de carga donde no varía el número de clientes concurrentes pero sí el número de peticiones realizadas, los tiempos tienen cierta igualdad. (Ver tabla 3)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| I.C.2 | I.C.3 | I.C.5 | I.C.7 | I.C.10 |
| 24,200 | 25,000 | 25,400 | 25,400 | 26,200 |

Encontramos, sin embargo, una excepción a la hora de realizar la primera petición a la página dinámica: el primer tiempo de respuesta es considerablemente alto comparado con los tiempos de las dos páginas estáticas. Esto puede deberse a que el servidor carga la página únicamente en la primera petición, generando ese pico de tiempo. Al permanecer cargada para siguientes peticiones, los tiempos de posteriores cargas no varían con respecto a los otros casos, dando como resultado tiempos similares.

Productividad

Al analizar la productividad, podemos ver que permanece indiferente tanto ante el número de peticiones como ante los tiempos de respuesta, encontrando incluso cierta estabilidad en cuanto a las peticiones atendidas por segundo según llegamos a las últimas intensidades de carga. Sin embargo, al atender a las medias de las pruebas como observamos en la figura, podemos ver tres caídas en la productividad: una en la primera petición de la página dinámica, y dos en la cuarta y sexta petición de la página estática pequeña.

(COMPARATIVA DE MEDIAS)

La caída en la página dinámica puede deberse, al igual que ocurre en el tiempo de respuesta, a la carga de la misma en esta primera llamada al servidor. Se puede ver que esta caída afecta incluso a la siguiente llamada, siendo algo más baja que el resto de peticiones. Esto explicaría que, en las siguientes peticiones, la productividad sea muy similar al resto de páginas de ahí en adelante.

La segunda y la tercera caída, que encontramos en la página estática pequeña, pueden deberse en un principio a una saturación en el servidor, algo que se respalda comparando estos datos con los tiempos de respuesta en esas mismas peticiones.

Encontramos, además, otros aspectos al analizar las gráficas de los datos individuales tanto en las páginas estáticas como en la dinámica. En estas, podemos ver que, a pesar de que las medias nos hagan pensar lo contrario, los datos son en general más dispersos de lo que cabría esperar.

En la figura , perteneciente a la página estática pequeña, observamos que la mayor dispersión de puntos se encuentra en las dos primeras peticiones al servidor, pudiendo ser explicado, al igual que la página dinámica, por la carga de la misma en estas primeras situaciones. Más tarde, podemos comprobar que la productividad en el resto de peticiones pierden esta dispersión, incluso en las caídas en la cuarta y sexta petición antes mencionadas.

(GRAFICA DE PEQUEÑO)

En la figura X, que muestra la productividad de la página estática grande, vemos que esta dispersión se encuentra en cada una de las peticiones al servidor. Es posible que, al ser una página más pesada, cualquier anomalía en la red interna de la UVa haga variar ampliamente la productividad de nuestro servidor.

(GRAFICA DE GRANDE)

Buscando las peticiones menos dispersas, encontramos solamente dos puntos en los que se igualan más los datos: la tercera y la duodécima petición. Pero incluso ocurriendo esto, vemos que en ambos la prueba 4 tiene una productividad notablemente menor que el resto de pruebas. Esto puede ser debido a que, a la hora de esta prueba, la red interna esté más saturada que el resto de horas.

Por último, en la figura X que nos muestra la gráfica de la página dinámica comprobamos que, a pesar de haber también dispersión, los datos que encontramos aquí son más estables por prueba, dando como resultado trazas invariables en la gráfica, sobre todo a partir de la séptima posición. La diferencia de peticiones por segundo entre las cinco pruebas puede deberse a las horas en que se han realizado y la carga de la red que esté soportando la red interna.

GRAFICA DE DINAMICO

SAR

Atendiendo a los usos de la CPU y sus porcentajes, podemos ver en qué momentos se producen las peticiones y cuánto tiempo duran, además de cuántos recursos del servidor utilizan.

En orden, primero realizamos las peticiones de la página estática pequeña, después las de la estática grande y después las de la dinámica. Teniendo en cuenta esto, vemos que las primeras peticiones a cada página, de menor intensidad de carga, ocupan menos recursos (un 7% en el porcentaje de user en la primera petición del estático en la primera prueba) y apenas duran unos segundos (2 segundos como máximo en la misma petición y misma prueba). Sin embargo, aquellas con mayor intensidad de carga utilizan porcentajes de CPU más altos (hasta el 50% de user y 47% de system en las peticiones de la página dinámica) y rondan los 25 segundos de uso de recursos.

Contrastando los porcentajes de uso de CPU con las intensidades de carga y desglosando estas en número de peticiones y en clientes concurrentes, podemos apreciar que los aumentos notables de tiempos y consumo de recursos se encuentran en los aumentos de peticiones. Se puede observar esto al comparar los recursos consumidos por diferentes intensidades de carga con el mismo número de peticiones, donde podemos ver que los porcentajes de uso de CPU son muy similares. (AÑADIR GRÁFICO DE EJEMPLO)

(Mostrar calculos del intervalo de confianza. Mostrar fórmula)

Respaldar análisis con SAR y apoyarse en las gráficas.

**Conclusiones**

Contrastando los datos obtenidos sobre los tiempos de respuesta, la productividad y el consumo de recursos en los diferentes escenarios, podemos apreciar diferentes observaciones:

* El número de clientes concurrentes afecta gravemente al tiempo de respuesta. Esto podría producir un cuello de botella en caso de atender a demasiados clientes concurrentes, retrasando ampliamente la responder de las peticiones recibidas.
* La productividad se ve directamente afectada por el peso de la página web. Esto se ve claramente al ver que las dispersiones más graves se encuentran en la página estática grande.
* Los recursos consumidos en el servidor son directamente proporcionales al número de peticiones recibidas. Entre user y system, llegan a ocupar casi la totalidad del uso del CPU, dejando el porcentaje ocioso en un 1% en algunas ocasiones.

(hablar sobre los datos atípicos. Posible referencia: <https://www.ibm.com/support/knowledgecenter/es/SS3RA7_sub/modeler_mainhelp_client_ddita/components/dt/ts_outliers_overview.html>)

**Bibliografía**

**Configuraciones**

Servidor

Para entrar en nuestra máquina lo primero que hacemos es buscar la dirección virtual que nos ha sido asignada, y que depende del número del grupo que somos, en nuestro caso el 22.

Para conectar mediante el ssh simplemente escribimos el siguiente comando:

$ ssh -p 31221 usuario@virtual.lab.inf.uva.es

Al entrar la primera vez en la máquina lo que hacemos es cambiar la contraseña:

$ sudo passwd

Una vez cambiada la contraseña comprobamos que todo está en orden y nos podemos conectar tanto por ssh como por X2GO, el cual nos permite de manera visual acceder a dicho servidor para poder empezar a configurar Apache.

Por último comprobamos la información de nuestra máquina virtual por si en un futuro nos pudiera ser útil:

$ sudo uname -a

$ Linux virtual 4.4.0-112-generic #135-Ubuntu SMP Fri Jan 19 11:48:36 UTC 2018 x86\_64 x86\_64 x86\_64 GNU/Linux

Apache

Como Apache ya estaba preinstalado en la máquina, no es necesario instalarlo. Sin embargo, si debemos modificar la opción para activar el modo KeepAlive, las conexiones **KeepAlive** son aquellas que se mantienen abiertas tras recibir una petición del cliente y servirla, de tal manera que la siguiente petición reaprovecha la conexión. \citar{<http://systemadmin.es/2008/11/conexiones-keepalive-de-apache>}

Para activarlo nos dirigimos al archivo /etc/apache2/apache2.conf, y modificamos el estado del KeepAlive a On.

$ sudo service apache2 restart

A continuación, necesitamos instalar la herramienta de monitorización SAR en el servidor. Dicha herramienta está incluida en el paquete Sysstat, por lo que procedemos a instalarlo mediante apt-get:

$ sudo apt-get install sysstat

Una vez realizadas ambas configuraciones reiniciamos apache mediante el siguiente comando para que surta efecto y tengamos el KeepAlive en funcionamiento y ya podamos usar SAR para visualizar el rendimiento del servidor.

**Anexo II - Casos de prueba.**

Se han determinado un total de 3 casos de prueba que representarán distintos escenarios:

* El primero será una página estática en html con una imagen menor de 50KB.
* El segundo será también una página estática pero con una imagen mayor de 250KB.
* El tercero será una página dinámica que esté ejecutando código.

En nuestro caso, la página dinámica consistirá en un código php que invoca una función, la cual, devuelve la fecha actual y lo muestre en la página.

<html>

<?php

$now = new DateTime();

echo $now -> format(‘Y-m-d H:i:s’);

?>

</html>

Para que sean accesibles desde fuera, movemos tanto el código de la página dinámica como las páginas estáticas a la carpeta donde se encuentra el servidor apache por defecto. Dicha carpeta tiene como ruta /var/www/html.