Practica 4: Benchmarking y Ajuste del Sistema

1. Phoronix Test (Benchmarks)

En esta practica vamos a tratar el tema de Benchmarking en nuestras máquina, para ello vamos hacer uso de Phoronix Test Suite, que es una suite en el que estan incluido numerosos benchmarkings. Lo primero que vamos a realizar es la instalación de Phoronix, para ellos haciendo uso de la documentacion[1] vamos a instalarlo, tan solo hay realizar como cualquier paquete en Ubuntu:

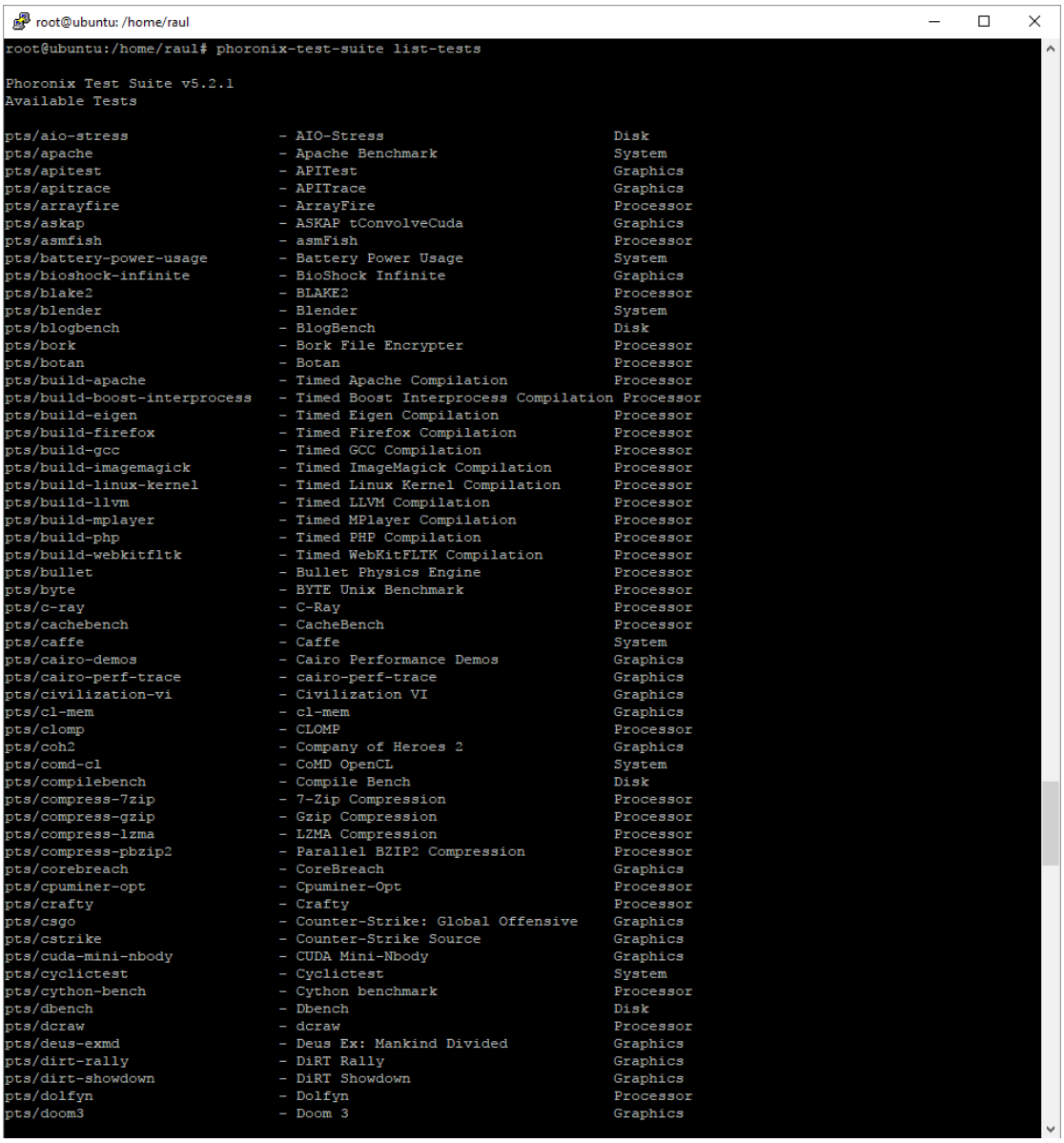
```
sudo apt-get install phoronix-test-suite
```

Ya tendriamos instalado Phoronix. Ahora para comprobar cuales son los benchmark que hay existentes en Phoronix podemos consultarlos a traves de la siguiente orden:

```
phoronix-test-suite list-tests
```

Al comprobar cuales son los benchmarks disponibles en phoronix, es decir, al realizar la orden anterior, nos pedira confirmación de los terminos de uso y nos pregunta si queremos o no enviar información anónima a modo de estadistica a Phoronix, podemos realizar lo que creamos conveniente.

Al comprobar los diferentes test que existen nos produce una salida como la siguiente:



Esa lista son todos los benchmarks disponibles hasta el momento en Phoronix, para ejecutar cualquier benchmark tan solo debemos ejecutarlo con la siguiente orden:

```
phoronix-test-suite run test
```

Donde la variable test habría que poner el test que queremos lanzar. Al ejecutar un test me he dado cuenta que me daba fallos por que no esta instalado php-zip, para instalarlo tan solo debemos realizar:

```
sudo apt-get install php-zip
```

Con esto ya podemos ejecutar cualquier test, para ellos vamos a probar el test "Sudokut", nos pedirá que si queremos instalarlo y le decimos que si. Para ejecutar el test Sudokut tenemos que realizar la orden que hemos dicho antes para ejecutar test, pero realizandola con el test sudokut.

```
phoronix-test-suite run pts/sudokut
```

A tener en cuenta que para realizar los benchmarks lo mejor es tener el sistema aislado lo máximo posible, con aislado me refiero a intentar no hacer nada con el sistema y tan simplemente ejecutar el benchmark, es decir, cerrar todos los programas, desconectar internet...
Al realizar el benchmark podemos observar lo siguientes resultados:

```
root@ubuntu:/home/raul# phoronix-test-suite run pts/sudokut

Phoronix Test Suite v5.2.1
System Information

Hardware:
Processor: Intel Core i5-5200U @ 2.19GHz (1 Core), Motherboard: Oracle VirtualBox v1.2, Chipset: Int
el 440FX- 82441FX PMC, Memory: 488MB, Disk: 2 x 9GB VBOX HDD, Graphics: InnoTek VirtualBox, Audio: I
ntel 82801AA AC 97 Audio, Network: Intel 82540EM Gigabit

Software:
OS: Ubuntu 16.04, Kernel: 4.4.0-87-generic (x86_64), File-System: ext4, Screen Resolution: 800x600,
System Layer: Oracle VirtualBox

    Would you like to save these test results (Y/n): n


Sudokut 0.4:
    pts/sudokut-1.0.0
    Test 1 of 1
    Estimated Trial Run Count:    3
    Estimated Time To Completion: 2 Minutes
        Started Run 1 @ 18:39:46
        Started Run 2 @ 18:40:11
        Started Run 3 @ 18:40:33  [Std. Dev: 3.39%]

    Test Results:
        19.402678012848
        19.109304189682
        18.175315856934

    Average: 18.90 Seconds
```

Podemos observar toda la información respectiva al test la cual empieza con el nombre del test y su versión que en este caso el test se llama "Sudokut" en su versión 0.4. Nos dice que el Test que realiza es 1 de 1 y que la estimación de la veces que va a ejecutar el test es de tres veces, esto quiere decir que el test de resolver un sudoku se va a realizar tres veces. Depués nos dice que el tiempo estimado de terminación del test es de 2 minutos. Seguidamente al tiempo estimado nos encontramos con tres lineas que ponen Started Run 1,2 y 3 esto significa que la primera ejecución se realizó a las 18:39:46, la segunda ejecución del test a las 18:40:11 y la tercera ejecución a las 18:40:33 que no son más que las horas del comienzo de los test. Cabe destacar una varianle muy importante que es "Std. Dev: 3.39%" esta varaible es la desviación estándar o desviación típica que no es más que el "promedio" o variación esperada con respecto a la media aritmética, es decir, como de dispersos están los datos con respecto a la media. Mientras mayor sea la desviación tipica más

dispersos están los datos y viceversa. Finalmente el test nos proporciona la media en tiempo (segundos) en los que mi procesador es capaz de realizar un test.

Si ejecutamos el mismo test la segunda vez, tendremos una media de ejecución menor y esto es debido a que todos los datos para la ejecución del programa ya estaban cargados anteriormente, luego ese tiempo que pierde el procesador en cargar datos en la primera ejecución ya no lo tiene que volver hacer, luego la media de ejecución del test baja. En la foto se puede apreciar la pequeña mejoría.

```
Sudokut 0.4:
    pts/sudokut-1.0.0
    Test 1 of 1
    Estimated Trial Run Count:    3
    Estimated Time To Completion: 2 Minutes
        Started Run 1 @ 18:54:14
        Started Run 2 @ 18:54:42
        Started Run 3 @ 18:55:03  [Std. Dev: 11.34%]
        Started Run 4 @ 18:55:23  [Std. Dev: 10.21%]
        Started Run 5 @ 18:55:43  [Std. Dev: 9.36%]
        Started Run 6 @ 18:56:03  [Std. Dev: 8.65%]

    Test Results:
        22.054853200912
        18.4430539608
        18.033337116241
        18.02688908577
        17.979267120361
        18.028002023697

    Average: 18.76 Seconds
```

Voy a ejecutar un segundo benchmark, en este caso va a ser el benchmark de encode-mp3, este benchmark calcula el tiempo medio en comprimir un archivo WAV a MP3, la ejecución nos ha dado una salida como la siguiente:

```
Hardware:
Processor: Intel Core i5-5200U @ 2.19GHz (1 Core), Motherboard: Oracle VirtualBox v1.2, Chipset: Int
el 440FX- 82441FX PMC, Memory: 488MB, Disk: 2 x 9GB VBOX HDD, Graphics: InnoTek VirtualBox, Audio: I
ntel 82801AA AC 97 Audio, Network: Intel 82540EM Gigabit

Software:
OS: Ubuntu 16.04, Kernel: 4.4.0-87-generic (x86_64), Compiler: GCC 5.4.0 20160609, File-System: ext4
, Screen Resolution: 800x600, System Layer: Oracle VirtualBox

    Would you like to save these test results (Y/n): n


LAME MP3 Encoding 3.99.5:
    pts/encode-mp3-1.6.0
    Test 1 of 1
    Estimated Trial Run Count:    5
    Estimated Time To Completion: 4 Minutes
        Started Run 1 @ 19:40:19
        Started Run 2 @ 19:40:42
        Started Run 3 @ 19:41:00
        Started Run 4 @ 19:41:19
        Started Run 5 @ 19:41:39  [Std. Dev: 3.50%]
        Started Run 6 @ 19:41:59  [Std. Dev: 3.31%]

    Test Results:
        16.800896883011
        16.872579813004
        16.967097997665
        17.753692150116
        18.15034198761
        16.874634027481

    Average: 17.24 Seconds
```

La salida es similar a la anterior solo varía que este benchmark realizar 6 test para realizar finalmente la media en la que el sistema es capaz de comprimir un archivo WAV a MP3. Inicialmente dice el test que va a realizar 5 veces la ejecución del test, pero esto de forma estimada por que como podemos observar finalmente realiza el test 6 veces, al igual que estima un tiempo total de 4 minutos para la ejecución del benchmark al completo y al final tarda menos.

2. Midiendo rendimiento servidor con ab y Jmeter

En esta segunda parte de la practica lo que vamos es a comprobar el rendimiento de nuestro servidor y ver que ocurre cuando a este le llegan muchas peticiones php, para ello primero vamos a realizar un primer procedimiento simple emitiendo peticiones a una maquina desde la otra y ver como reacciona, para ello usaremos el comando "ab", finalmente realizaremos lo mismo pero con una herramienta más potente como es "Jmeter".

El comando "ab" ya está instalado al haber realizado la instalación de apache en nuestro servidor, luego tan solo debemos usarlo desde una maquina a otra para comprobar el rendimiento de estas ante tantas peticiones. Para realizar este procedimiento lo hacemos de la siguiente forma:

```
ab -n 10000 -c 1 -g datos.csv 192.168.56.110/
```

En la dirección ip tenemos que poner la ip de la maquina a la que queremos enviar las peticiones, en este caso estoy realizando el envio de peticiones desde Ubuntu Server a CentOS, si queremos hacerlo al revés tan solo habría que cambiar la ip por la de Ubuntu Server. La explicación del comando anterior es que -n hace referencia al numero de peticiones que se van a realizar y -c es la concurrencia, es decir, que si ponemos concurrencia de 1 quiere decir que va a ir emitiendo solo 1 peticion cada vez, si ponemos otro valor mayor lo va a realizar con esa concurrencia, es decir, que se van a enviar de dos en dos o de tres en tres o el numero que hace referencia a la concurrencia. En la maquina virtual no tiene sentido ponerle una concurrencia mayor a 1 ya que solo tenemos un procesador y una sola hebra.
Para comprobar cuantas hebras activas utiliza el comando ab, podemos realizar la orden ab pero poniendole una concurrencia por ejemplo de 5 o 10 y mientras esta enviando peticiones realizamos la siguiente orden:

```
ps -A | grep ab | wc -l
```

La cual nos va a devolver el numero de ejecuciones a la vez de ab, lo que equivale al numero de hebras totales que están trabajando. Dejo capturas del resultado al realizar "ab" contra CentOS desde Ubuntu Server y viceversa, el primer caso es desde Ubuntu a CentOS y elo segundo caso de CentOS a Ubuntu. Los datos que nos aparecen son los siguientes:

Server Software: Versión y tipo de servidor instalado
Server Hostname: Ip del servidor al que enviamos peticiones
Server Port: Puerto del servidor
Document Path: Pathname del documento, en este caso es el directorio principal
Document Length: Tamaño que tiene que descargar cada vez que enviamos una petición
Concurrency level: Nivel de concurrencia (en nuestro caso 1 por lo que explicado anteriormente)
Time taken for tests: Tiempo en el que las peticiones se satisfacen correctamente
Complete request: Numero de peticiones totales
Failed request: Numero de peticiones que han fallado
Write errors: Errores en la escritura
Total transferred: Numero de bytes totales transferidos
HTML transferred: Numero de bytes HTML transferidos
Request per second: Numero de peticiones por segundo
Time per request: Tiempo por petición enviada
Transfer rate: Ratio de transferencia

Luego en cuando a la explicación a la ejecutción de "ab" contra CentOS podemos decir que, para 1000 peticiones con una concurrencia igual a 1, el resultado es que todas las peticiones se satisfacen correctamente en 1.211 segundos. El tiempo por cada petición es de 1.211 milisegundos. En cambio si vemos la

información del servidor de Ubuntu podemos observar que para 1000 peticiones con una concurrencia igual a 1, el resultado es que todas las peticiones se satisfacen correctamente en 2.164 segundos. El tiempo por cada petición es de 2.164 milisegundos. Como podemos observar, aparentemente una maquina es más rapida que otra, pero no solo depende de la rapidez de la máquina, si no también de otras variables como la cantidad de bytes de descarga que es diferente, depende de la red...

```
root@ubuntu:/home/raul# ab -n 1000 -c 1 192.168.56.110/
This is ApacheBench, Version 2.3 <$Revision: 1706008 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.56.110 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:        Apache/2.4.6
Server Hostname:        192.168.56.110
Server Port:            80

Document Path:          /
Document Length:        4897 bytes

Concurrency Level:      1
Time taken for tests:   1.211 seconds
Complete requests:      1000
Failed requests:        0
Non-2xx responses:      1000
Total transferred:      5179000 bytes
HTML transferred:       4897000 bytes
Requests per second:    826.07 [#/sec] (mean)
Time per request:       1.211 [ms] (mean)
Time per request:       1.211 [ms] (mean, across all concurrent requests)
Transfer rate:          4177.95 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       1
Processing:     1    1   0.1      1       2
Waiting:        0    1   0.1      1       1
Total:          1    1   0.2      1       3

Percentage of the requests served within a certain time (ms)
  50%      1
  66%      1
  75%      1
  80%      1
  90%      1
  95%      1
  98%      2
  99%      2
 100%      3 (longest request)
```

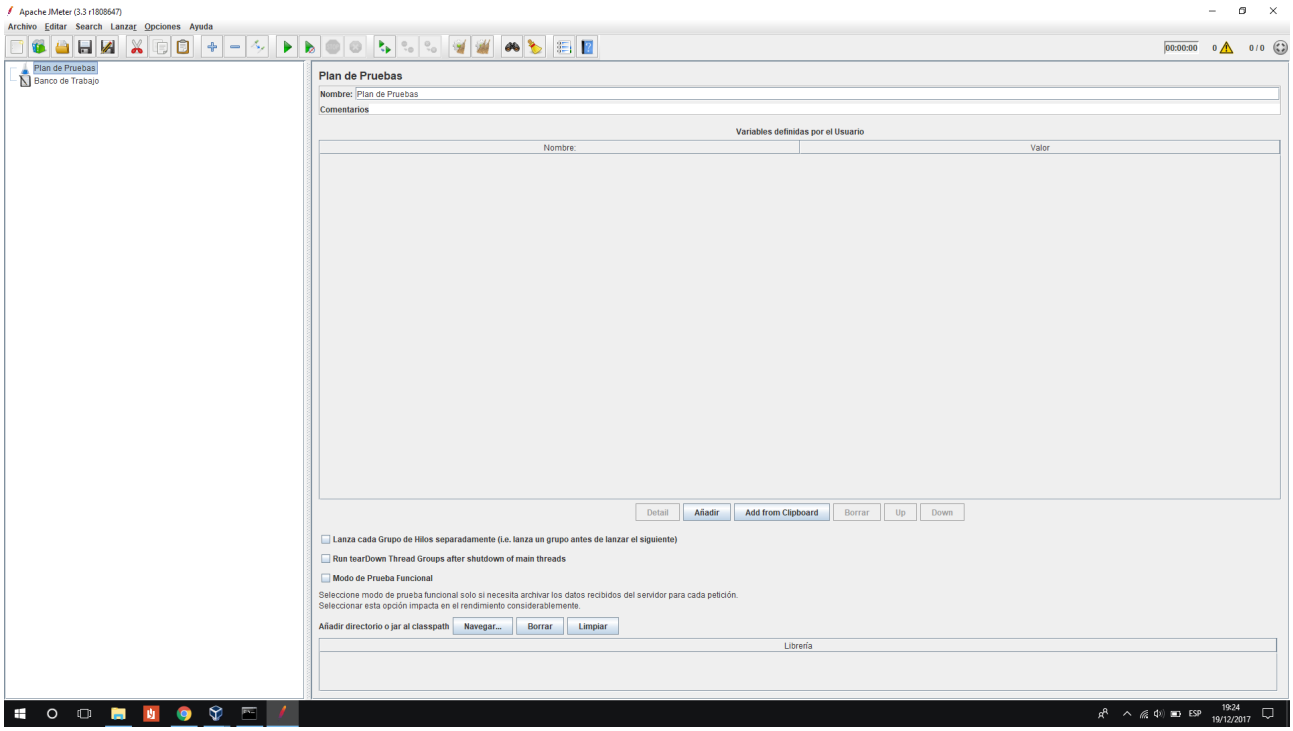
```
[root@localhost ~]# ab -n 1000 -c 1 192.168.56.105/
This is ApacheBench, Version 2.3 <$Revision: 1430300 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.56.105 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:        Apache/2.4.18
Server Hostname:        192.168.56.105
Server Port:            80

Document Path:          /
Document Length:        11321 bytes

Concurrency Level:      1
Time taken for tests:   2.164 seconds
Complete requests:      1000
Failed requests:        0
Write errors:           0
Total transferred:      11595000 bytes
HTML transferred:       11321000 bytes
Requests per second:    462.02 [#/sec] (mean)
Time per request:       2.164 [ms] (mean)
Time per request:       2.164 [ms] (mean, across all concurrent requests)
Transfer rate:          5231.53 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    1   0.3      1       2
Processing:     1    1   0.4      1       6
Waiting:        1    1   0.3      1       5
Total:          1    2   0.5      2       6

Percentage of the requests served within a certain time (ms)
  50%      2
  66%      2
  75%      2
  80%      2
  90%      3
  95%      3
  98%      3
  99%      4
 100%      6 (longest request)
```

Ahora voy a realizar el mismo experimento, pero realizando las peticiones desde el host a ambas máquinas, en mi caso como estoy en Windows corriendo ambas máquinas, he tenido que descargar los binarios de apache y sacar el ejecutable de ab para poderlo ejecutar desde el CMD. Mismo esquema que antes, la primera captura es a Ubuntu y la segunda a CentOS, todo desde el host (anfitrión). Esta vez la concurrencia si es de 4 por que como podemos observar la variable "Time per request" el total en Ubuntu han sido 14.621milisegundos de los cuales cada peticion se realiza en una media de 3.655 milisegundos, en cambio en CentOS tarda 13.432 milisegundos de los cuales cada petición se realiza en una media de 3.358 milisegundos. Luego si tenemos en cuenta solo este aspecto podemos decir que CentOS es algo más rapido, pero si observamos son diferentes tamaños de carga y la velocidad de transferencia es mayor en Ubuntu, luego en este caso podríamos decir que Ubuntu es más rapido, luego es bastante complicado definir cual es el servidor más rapido debido a que existen muchas variables que hay que tener en cuenta.

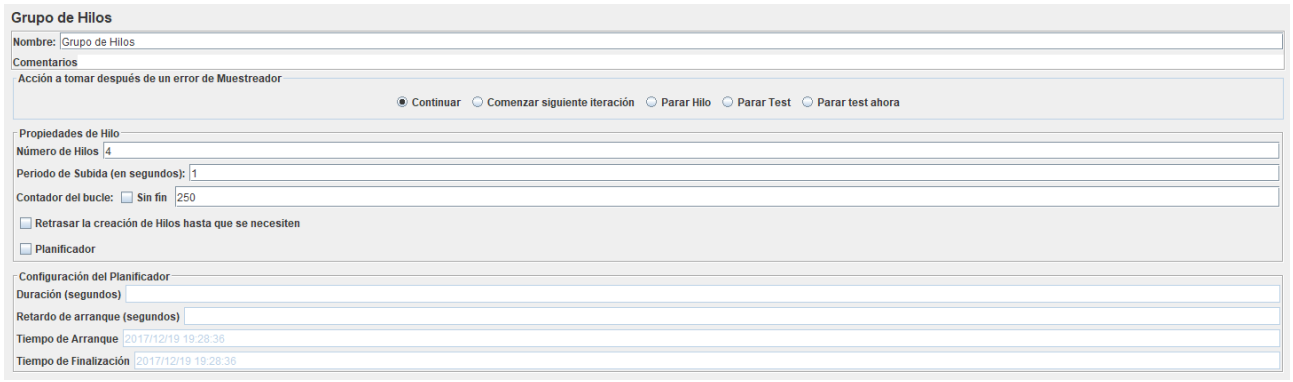
```
C:\>ab -n 1000 -c 4 http://192.168.56.105/
This is ApacheBench, Version 2.3 <$Revision: 1807734 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.56.105 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:        Apache/2.4.18
Server Hostname:        192.168.56.105
Server Port:            80

Document Path:          /
Document Length:        11321 bytes

Concurrency Level:      4
Time taken for tests:   3.655 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      11595000 bytes
HTML transferred:       11321000 bytes
Requests per second:    273.58 [#/sec] (mean)
Time per request:       14.621 [ms] (mean)
Time per request:       3.655 [ms] (mean, across all concurrent requests)
Transfer rate:          3097.77 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    1   0.6      1       4
Processing:     5   14   4.2     14      35
Waiting:        2   12   5.2     13      35
Total:          6   14   4.2     15      37

Percentage of the requests served within a certain time (ms)
  50%     15
  66%     16
  75%     17
  80%     18
  90%     20
  95%     22
  98%     24
  99%     28
 100%     37 (longest request)
```

```
C:\>ab -n 1000 -c 4 http://192.168.56.110/
This is ApacheBench, Version 2.3 <$Revision: 1807734 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.56.110 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:        Apache/2.4.6
Server Hostname:        192.168.56.110
Server Port:            80

Document Path:          /
Document Length:        4897 bytes

Concurrency Level:      4
Time taken for tests:   3.358 seconds
Complete requests:      1000
Failed requests:        0
Non-2xx responses:      1000
Total transferred:      5179000 bytes
HTML transferred:       4897000 bytes
Requests per second:    297.79 [#/sec] (mean)
Time per request:       13.432 [ms] (mean)
Time per request:       3.358 [ms] (mean, across all concurrent requests)
Transfer rate:          1506.09 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    1   0.6      1       4
Processing:     5   12   3.0     13      22
Waiting:        3   11   3.9     12      22
Total:          6   13   3.0     14      23

Percentage of the requests served within a certain time (ms)
  50%     14
  66%     15
  75%     16
  80%     16
  90%     17
  95%     19
  98%     20
  99%     21
```

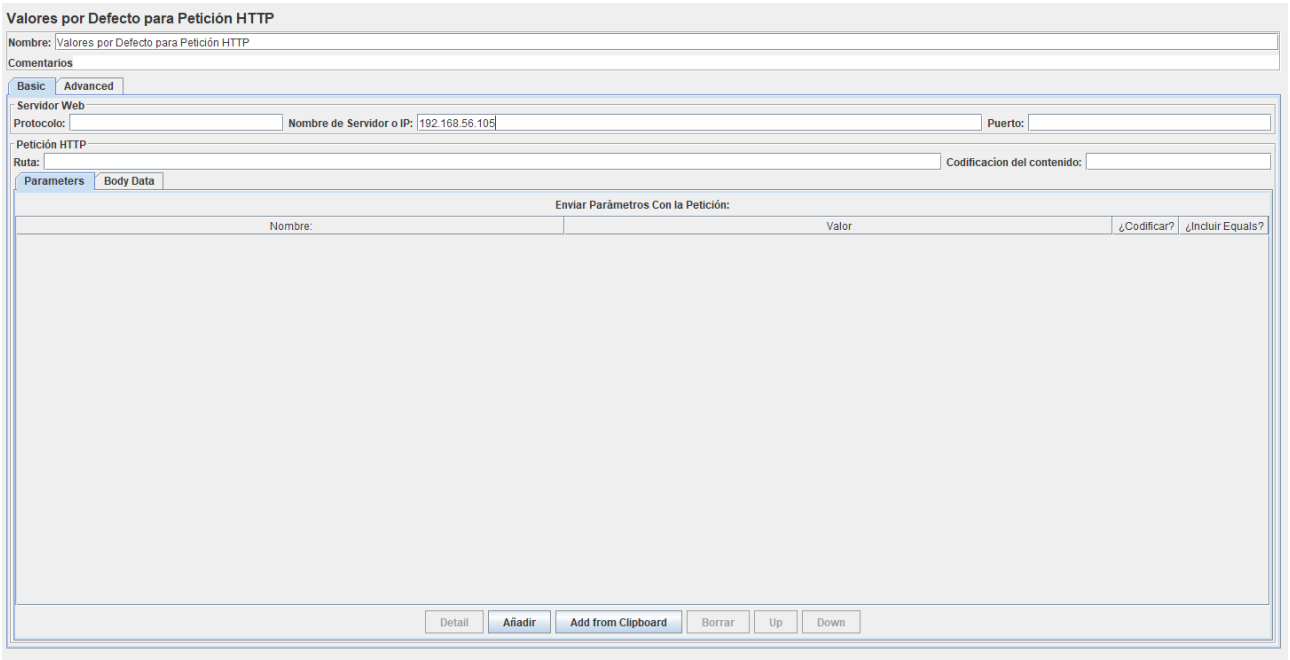
Para finalizar la practica vamos a comprobar lo mismo que comprobamos con "Apache Benchmark" el rendimiento de nuestro servidor, pero esta vez usando Jmeter, que es una herramienta para aplicar cargas a un servidor y comprobar el rendimiento de este cuando lo estresamos. Lo primero que vamos hacer es instalarlo en nuestro Host, en mi caso Windows 10 y por lo tanto he accedido a la web de Jmeter y he descargado los binarios para poder ejecutarlo. Hay que tener en cuenta que debemos tener instalado Java en su versión 8 para que esta herramienta funciones, una vez instalada tendrémos una interfaz como la siguiente:



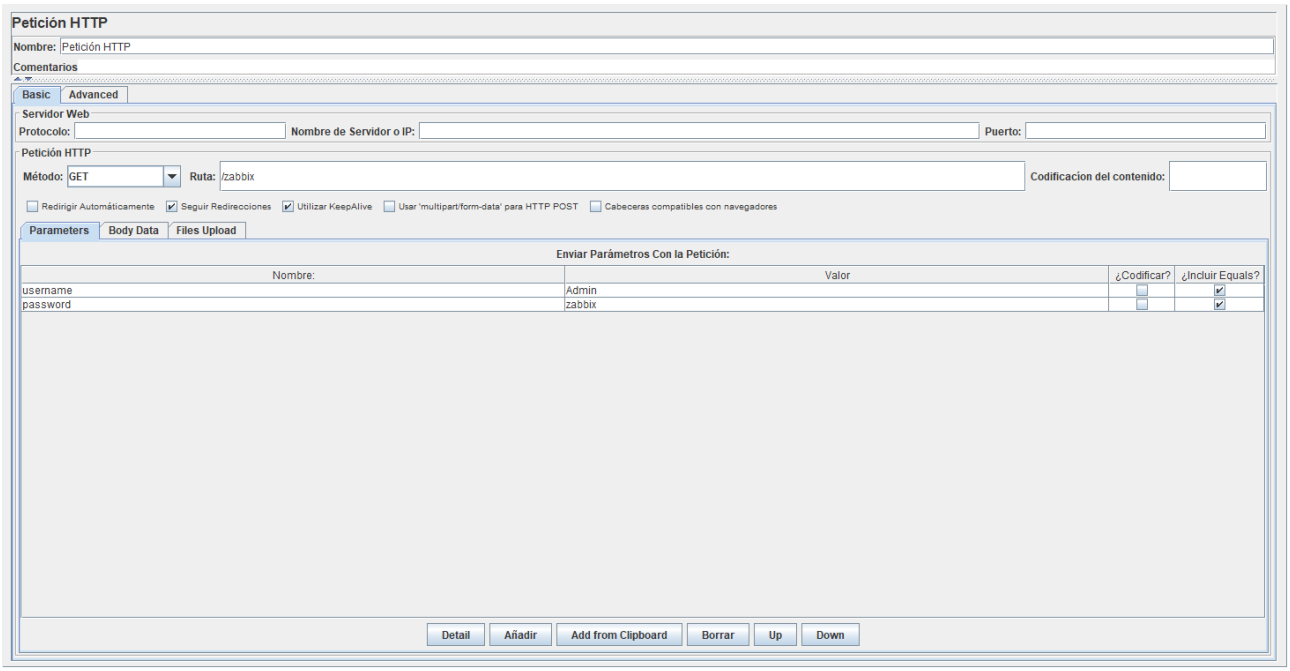
Una vez abierta la interfaz creamos un "Grupo de Hilos", lo cual sirve para indicar la concurrencia. Para ello en "Plan de Pruebas" pulsamos boton derecho y realizamos click en "Añadir-Hilos(Usuarios)-Grupo de hilos", seguidamente le damos al "Grupo de Hilos" creado y en "Numero de hilos" ponemos 4, ponemos 4 para realizar el mismo procedimiento que hicimos con "ab", luego en "Contador del bucle" ponemos 250, para realizar un total de 1000 peticiones a nuestro servidor.



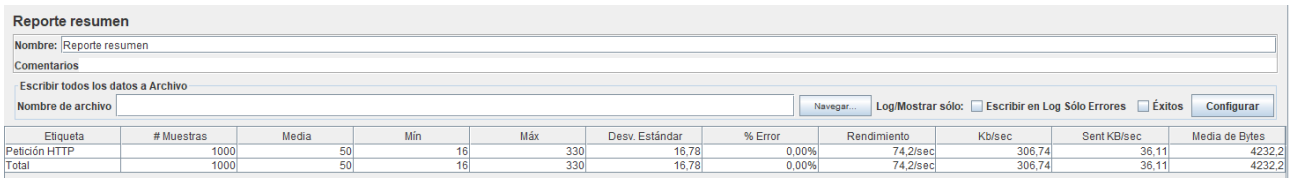
Lo siguiente que vamos a configurar es la petición HTTP que tiene por defecto Jmeter. Para ello tenemos que seleccionar "Grupo de Hilos" con el botón derecho y realizamos "Añadir-Elemento de Configuración-Valores por Defecto para Petición HTTP" y configuramos de la siguiente manera:



Luego añadimos lo siguiente "Añadir-Muestreador-Petición HTTP" e incluimos /zabbix y el usuario y contraseña de zabbix, tal y como aparece en la captura.

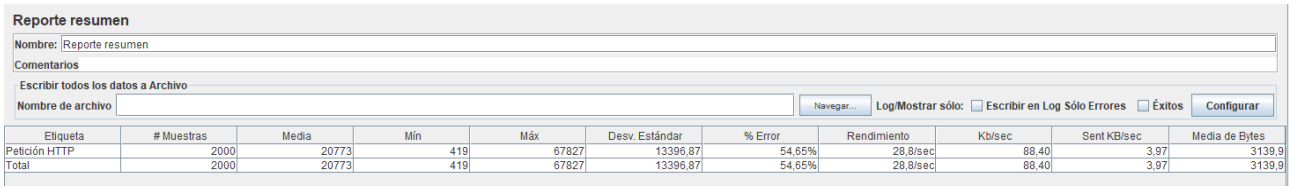


Añadimos el item "Añadir-Receptor-Gráfico de Resultados".
Finalmente agregamos un item para obtener más información el cual es "Añadir-Receptor-Reporte resumen"
Una vez que hemos realizado todo este procedimiento tan solo debemos darle al simbolo de "Play" verde que hay en la barra superior y se iniciará la prueba. Al final de la prueba podemos ver como en "Grafico de Resultados" nos produce un gráfico como el siguiente:



Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Bytes
Petición HTTP	1000	50	16	330	16,78	0,00%	74,2/sec	306,74	36,11	4232,2
Total	1000	50	16	330	16,78	0,00%	74,2/sec	306,74	36,11	4232,2

Como podemos observar en el gráfico se puede ver representado el rendimiento, la media, la mediana, la desviación tipica y los datos obtenidos. Podemos observar como el servidor (UBUNTU) es capaz de procesar hasta 4.453,021/minuto si mantiene el mismo rendimiento para ese número de peticiones.
Ahora para comprobar cuanto puedo llegar a estresar el servidor voy a poner en "Grupo de Hilos" como numero de hilos 2000 y en contador del bucle 1, para que 2000 usuarios realicen una sola petición. Los resultados al ejecutarlo es el siguiente:



Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Bytes
Petición HTTP	2000	20773	419	67827	13396,87	54,65%	28,8/sec	88,40	3,97	3139,9
Total	2000	20773	419	67827	13396,87	54,65%	28,8/sec	88,40	3,97	3139,9

Al realizar esta prueba podemos ver que al realizar todas las peticiones se ha producido un error del 54,65% de las peticiones enviadas eso quiere decir que solo se han enviado correctamente 45,35% de las peticiones. Luego podemos decir que nuestro servidor no es capaz de resistir el login en Zabbix de 2000 usuarios a la vez. Tendriamos que realizar pruebas con el numero de usuarios que suelen entrar al servidor para asimilarlo al maximo con la realidad.

BIBLIOGRAFIA

[1] Documentación Phoronix: https://www.phoronix-test-suite.com/documentation/phoronix-test-suite.html#InstallationInstructions
[2] Desviación típica: https://support.minitab.com/es-mx/minitab/18/help-and-how-to/statistics/basic-statistics/supporting-topics/data-concepts/what-is-the-standard-deviation/
[3] Descarga Jmete: https://archive.apache.org/dist/jmeter/binaries/