

INGENIERÍA DE SERVIDORES (2016-2017)
DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y MATEMÁTICAS
UNIVERSIDAD DE GRANADA

Memoria Práctica 4

Simón López Vico

16 de mayo de 2017

Índice

1. Seleccione, instale y ejecute uno, comente los resultados. Atención: no es lo mismo un benchmark que una suite, instale un benchmark.	4
2. De los parámetros que le podemos pasar al comando, ¿qué significa -c 5? ¿Y -n 100? Monitorice la ejecución de ab contra alguna máquina (cualquiera). ¿Cuántas “tareass” crea ab en el cliente?	5
3. Ejecute ab contra a las tres máquinas virtuales (desde el SO anfitrión a las máquinas virtuales de la red local) una a una (arrancadas por separado). ¿Cuál es la que proporciona mejores resultados? Muestre y coméntelos. Use como máquina de referencia Ubuntu Server para la comparativa.	6
4. Instale y siga el tutorial en http://jmeter.apache.org/usermanual/build-web-test-plan.html realizando capturas de pantalla y comentándolas. En vez de usar la web de JMeter, haga el experimento usando sus máquinas virtuales. ¿Coincide con los resultados de ab?	8
5. Programe un benchmark usando el lenguaje que desee o busque uno ya programado (github, openbenchmark, etc.). Debe especificar: 1) Objetivo del benchmark. 2) Métricas (unidades, variables, puntuaciones, etc.). 3) Instrucciones para su uso. 4) Ejemplo de uso.	11
5.1. Tenga en cuenta que puede comparar varios gestores de BD, lenguajes de programación web (tiempos de ejecución, gestión de memoria, ...), duración de la batería, servidor DNS, etc. Haga tres ejecuciones del mismo, cambie al menos uno de los parámetros de la máquina virtual y vuelva a ejecutar ¿es, estadísticamente hablando, significativa la diferencia entre una configuración y otra?	13

Índice de figuras

1.1. Lista de test recomendados por Phoronix Suite.	4
1.2. Configuración previa al inicio del test.	4
2.1. Número de tareas que genera ab.	6
3.1. Resultados de ab sobre Ubuntu Server.	6
3.2. Resultados de ab sobre CentOS.	7
3.3. Resultados de ab sobre Windows Server 2008R2.	7
4.1. Asignación de las propiedades del grupo de hilos.	8
4.2. Escribiendo la IP de la máquina sobre la que se realizará el test.	9
4.3. Ruta de la máquina sobre la que realizaremos las peticiones HTTP.	9

4.4. Gráfica con los resultados del test.	10
4.5. Reporte del resumen generado tras el test.. . . .	10
5.1. Monitorización de la CPU con el benchmark <code>collectl</code>	12
5.2. Monitorización de la red de Internet con el benchmark <code>collectl</code>	13
5.3. Monitorización de la CPU lanzado procesos entre medias.	13
5.4. Reducción del número de CPUs de la máquina virtual.	14
5.5. Monitorización con una única CPU.	14

Índice de tablas

1. Seleccione, instale y ejecute uno, comente los resultados. Atención: no es lo mismo un benchmark que una suite, instale un benchmark.

Primero instalaremos Phoronix Suite mediante el comando `sudo apt install phoronix-test-suite`, y tras ello listaremos el conjunto de tests recomendados mediante `phoronix-test-suite list-recommended-tests`.

```
simon@simon-VirtualBox:~$ phoronix-test-suite list-recommended-tests
Phoronix Test Suite v4.8.3
Recommended OpenBenchmarking.org Test Profiles

Graphics Tests
pts/xonotic-1.5.0      - Xonotic
pts/etlegacy-1.0.0    - ET: Legacy
pts/openarena-1.5.3   - OpenArena
pts/gputest-1.3.1     - GpuTest
pts/bioshock-infinite-1.0.1 - BioShock Infinite
pts/apitest-1.1.0     - APITest
pts/cairo-demos-1.0.1 - Cairo Performance Demos

System Tests
pts/apache-1.6.1      - Apache Benchmark
pts/phpbench-1.1.1    - PHPBench
system/darktable-1.0.0 - Darktable
pts/phpbench-1.1.0    - PHPBench
pts/battery-power-usage-1.0.0 - Battery Power Usage
pts/nginx-1.1.0       - NGINX Benchmark
pts/tjbench-1.0.1     - libjpeg-turbo tjbench

Processor Tests
pts/c-ray-1.1.0       - C-Ray
pts/cachebench-1.1.0  - CacheBench
pts/build-linux-kernel-1.7.0 - Timed Linux Kernel Compilation

pts/scimark2-1.2.1    - SciMark
pts/compress-7zip-1.6.2 - 7-Zip Compression
pts/scimark2-1.2.0    - SciMark
pts/compress-gzip-1.1.0 - Gzip Compression

Disk Tests
pts/aio-stress-1.1.1  - AIO-Stress
pts/unpack-linux-1.0.0 - Unpacking The Linux Kernel
pts/hdparm-read-1.0.0 - hdparm Titled Disk Reads
pts/aio-stress-1.1.0  - AIO-Stress
pts/tiobench-1.1.0    - Threaded I/O Tester
pts/lozone-1.7.0      - IOzone
pts/fio-1.1.0         - Flexible IO Tester

Memory Tests
pts/stream-1.3.1      - Stream
pts/ramspeed-1.4.0    - RAMspeed SMP
pts/stream-1.2.0      - Stream
pts/t-test1-1.0.0     - t-test1
pts/stream-1.1.0      - Stream

Network Tests
pts/network-loopback-1.0.1 - Loopback TCP Network Performance
pts/network-loopback-1.0.0 - Loopback TCP Network Performance

simon@simon-VirtualBox:~$
```

Figura 1.1: Lista de test recomendados por Phoronix Suite.

Elegiremos un test para comprobar la velocidad de nuestra memoria RAM, y lo instalaremos mediante `sudo phoronix-test-suite install-test pts/ramspeed-1.4.0`. A continuación, ejecutaremos el test y se nos preguntará sobre qué acciones queremos realizar las comprobaciones (*copy*, *scale*, *add*, *triad*, *average* o *test all options*). Elegiremos comprobar la velocidad media de nuestra memoria RAM, y seleccionaremos que nos devuelva la solución en coma flotante.

```
simon@simon-VirtualBox:~$ phoronix-test-suite run pts/ramspeed-1.4.0

RAMspeed SMP 3.5.0:
pts/ramspeed-1.4.0
Memory Test Configuration
 1: Copy
 2: Scale
 3: Add
 4: Triad
 5: Average
 6: Test All Options
Type: 5

 1: Integer
 2: Floating Point
 3: Test All Options
Benchmark: 2
```

Figura 1.2: Configuración previa al inicio del test.

Finalmente, se nos dará la opción de guardar el resultado en un fichero (creado automáticamente) y nos aparecerá el tiempo estimado para que finalice el test. Tras completarse, se imprimirá por pantalla el siguiente resultado:

```
Test 1 of 1
Estimated Trial Run Count:    1
Estimated Time To Completion: 6 Minutes
    Started Run 1 @ 12:51:57
```

```
Test Results:
    8449.67
```

```
Average: 8449.67 MB/s
```

Por tanto, la velocidad media de nuestra memoria RAM será de unos 8 GB por segundo.

2. De los parámetros que le podemos pasar al comando, ¿qué significa `-c 5`? ¿Y `-n 100`? Monitoree la ejecución de `ab` contra alguna máquina (cualquiera). ¿Cuántas “tareas” crea `ab` en el cliente?

Si consultamos la entrada de `ab` en el manual de Linux[1], veremos que la opción `-c` [*concurrency*] sirve para elegir el número de peticiones múltiples que podemos realizar a la vez, siendo por defecto 1; en el caso del enunciado, `ab -c 5` significará que se mandarán las peticiones de 5 en 5.

Por otra parte, `-n` [*request*] establecerá el número de solicitudes a enviar durante el *benchmarking*, y al igual que con `-c`, el número por defecto es 1, lo que puede conducir a resultados no representativos. En el caso del enunciado, se enviarán 100 solicitudes para la ejecución del benchmark. Por tanto, con `ab -c 5 -n 100` se realizarán 100 solicitudes de 5 peticiones cada una.

Ahora, ejecutaremos el comando `ab` con los parámetros del enunciado desde Ubuntu a CentOS mediante la orden:

```
ab -c 5 -n 999999 http://192.168.56.101/ &
```

donde estableceremos 999999 peticiones para que tarde en ejecutarse del todo y nos de tiempo a comprobar cuantas “tareas” se han creado mediante el comando `ab`. Además, añadiremos `&` al final del comando para que se ejecute en segundo plano.

Para comprobar el número de tareas, utilizaremos el comando `ps` [2] que nos imprime por pantalla los procesos activos en el momento de ejecución, y lo encauzaremos con `grep ab` [3] para que nos muestre todas las tareas relacionadas con `ab`.

```

simon@simon-VirtualBox:~$ ab -c 5 -n 999999 http://192.168.56.101/ &
[1] 4339
simon@simon-VirtualBox:~$ This is ApacheBench, Version 2.3 <$Revision: 1528965 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.56.101 (be patient)

simon@simon-VirtualBox:~$ ps | grep ab
4339 pts/0    00:00:03 ab

```

Figura 2.1: Número de tareas que genera ab.

Finalmente, podemos ver que el comando ab solamente generará una tarea durante su ejecución en el cliente.

3. Ejecute ab contra a las tres máquinas virtuales (desde el SO anfitrión a las máquina virtuales de la red local) una a una (arrancadas por separado). ¿Cuál es la que proporciona mejores resultados? Muestre y coméntelos. Use como máquina de referencia Ubuntu Server para la comparativa.

Pondremos todas las máquinas en modo “*Host Only*”, modificaremos la página por defecto del servidor web de cada una de las máquinas y realizaremos el comando `ab -c 10 -n 1000 http://[ip_de_la_máquina]/`, obteniendo los siguientes resultados.

■ Ubuntu Server:

El test realizado sobre Ubuntu Server ha consumido 1.219 segundos, y podemos adelantar que ha sido el más rápido de los 3 tests realizados. Se han completado las 1000 peticiones sin ningún error, y cada una de esas peticiones concurrentes (de 10 en 10) ha tardado 12.193 milisegundos de media, por lo que cada petición ha consumido 1.219 milisegundos. La velocidad de transferencia ha sido de 95066.04 Kbytes/segundo, realizándose 820.13 peticiones por segundo de media.

Server Software:	Apache/2.4.10	Connection Times (ms)				
Server Hostname:	192.168.56.103		min	mean[+/-sd]	median	max
Server Port:	80	Connect:	0	1 1.6	1	17
Document Path:	/	Processing:	4	11 4.4	10	60
Document Length:	118422 bytes	Waiting:	0	5 3.2	5	29
		Total:	5	12 4.5	11	61
Concurrency Level:	10	Percentage of the requests served within a certain time (ms)				
Time taken for tests:	1.219 seconds	50%	11			
Complete requests:	1000	66%	13			
Failed requests:	0	75%	14			
Total transferred:	118698000 bytes	80%	15			
HTML transferred:	118422000 bytes	90%	17			
Requests per second:	820.13 [#/sec] (mean)	95%	20			
Time per request:	12.193 [ms] (mean)	98%	25			
Time per request:	1.219 [ms] (mean, across all concurrent requests)	99%	29			
Transfer rate:	95066.04 [Kbytes/sec] received	100%	61 (longest request)			

Figura 3.1: Resultados de ab sobre Ubuntu Server.

- **CentOS:**

La máquina CentOS es la que más tiempo ha consumido en la realización de los tests, con 1.309 segundos. Como podemos ver en la imagen 3.2, la longitud del documento es un poco menor que la longitud de la página de Ubuntu Server, aunque no será un valor realmente significativo. Las peticiones concurrentes de CentOS han tardado 13.090 milisegundos, por lo que cada una ha tardado 1.309 milisegundos. Por otra parte, se realizarán 763.96 peticiones por segundo y la velocidad de transferencia en el test de CentOS será de 88545.03 Kbytes/segundo.

Server Software:	Apache/2.4.6	Connection Times (ms)				
Server Hostname:	192.168.56.104		min	mean[+/-sd]	median	max
Server Port:	80	Connect:	0	1 1.0	1	14
Document Path:	/	Processing:	5	12 3.4	11	25
Document Length:	118418 bytes	Waiting:	2	7 2.7	6	22
		Total:	6	13 3.5	12	27
Concurrency Level:	10	Percentage of the requests served within a certain time (ms)				
Time taken for tests:	1.309 seconds	50%	12			
Complete requests:	1000	66%	14			
Failed requests:	0	75%	15			
Total transferred:	118685000 bytes	80%	15			
HTML transferred:	118418000 bytes	90%	17			
Requests per second:	763.96 [#/sec] (mean)	95%	19			
Time per request:	13.090 [ms] (mean)	98%	23			
Time per request:	1.309 [ms] (mean, across all concurrent requests)	99%	24			
Transfer rate:	88545.03 [Kbytes/sec] received	100%	27 (longest request)			

Figura 3.2: Resultados de ab sobre CentOS.

- **Windows Server 2008R2:** Windows Server ha consumido un total de 1.280 segundos para realizar el test. Notaremos que la longitud del documento asociado a Windows Server es unos 4000 bytes mayor que el de Ubuntu Server, aún siendo exactamente el mismo; esto es debido a que el formato de codificación es diferente entre las dos máquinas y el documento de Windows aumentará su tamaño respecto a las máquinas Linux usadas.

El tiempo de cada petición concurrente es de 12.799 milisegundos, y de las peticiones únicas de 1.280 milisegundos. Se habrán realizado de media 781.31 peticiones por segundo, con una velocidad de transferencia de 93917.88 Kbytes/segundo.

Server Software:	Microsoft-IIS/7.5	Connection Times (ms)				
Server Hostname:	192.168.56.102		min	mean[+/-sd]	median	max
Server Port:	80	Connect:	0	1 1.2	1	18
Document Path:	/	Processing:	3	11 2.4	11	21
Document Length:	122845 bytes	Waiting:	0	4 2.4	4	19
		Total:	4	13 2.6	12	27
Concurrency Level:	10	Percentage of the requests served within a certain time (ms)				
Time taken for tests:	1.280 seconds	50%	12			
Complete requests:	1000	66%	13			
Failed requests:	0	75%	14			
Total transferred:	123091000 bytes	80%	15			
HTML transferred:	122845000 bytes	90%	16			
Requests per second:	781.31 [#/sec] (mean)	95%	17			
Time per request:	12.799 [ms] (mean)	98%	19			
Time per request:	1.280 [ms] (mean, across all concurrent requests)	99%	21			
Transfer rate:	93917.88 [Kbytes/sec] received	100%	27 (longest request)			

Figura 3.3: Resultados de ab sobre Windows Server 2008R2.

Con los datos obtenidos, podemos concluir que la máquina con mejores resultados sería Ubuntu Server, seguida de Windows Server 2008R2 y finalmente CentOS. Aún así, estos resultados no son totalmente exactos, pues los tests realizados dependen de muchas variables que son distintas para cada uno de los servidores, y los valores de los 3 tests son

bastante similares, por lo que puede que en alguna otra realización del benchmark los datos de Windows salieran mayores a los de el resto, por ejemplo.

4. Instale y siga el tutorial en <http://jmeter.apache.org/usermanual/build-web-test-plan.html> realizando capturas de pantalla y comentándolas. En vez de usar la web de JMeter, haga el experimento usando sus máquinas virtuales. ¿Coincide con los resultados de ab?

Comenzaremos instalado JMeter desde la terminal con el comando `sudo apt install jmeter`. Tras esto, simplemente tendremos que usar el comando `jmeter` para ejecutar la interfaz gráfica del programa.

El primer paso que llevaremos a cabo será crear un grupo de hilos, haciendo click derecho sobre “*Plan de pruebas*” y seleccionando *Añadir/Hilos (Usuarios)/Grupo de Hilos*. Automáticamente se nos abrirá la ventana respectiva al grupo de hilos, donde le daremos un nombre (*prueba*) y escogeremos el número de hilos (usuarios) a usar, así como el período de subida y el número de veces que se realizarán peticiones. En nuestro caso escogeremos 1 usuario (no 10, ya que el `-c` en `ab` especifica la concurrencia, no el número de usuarios) y 1000 repeticiones, para más tarde poder comparar con el test obtenido mediante `ab`. (`ab -c 10 -n 1000`).

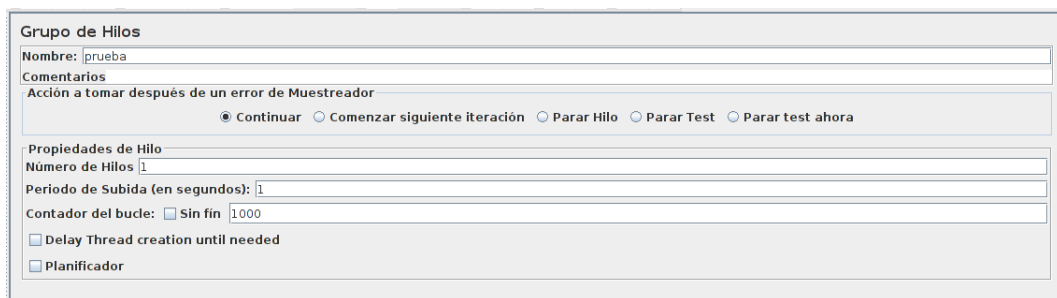


Figura 4.1: Asignación de las propiedades del grupo de hilos.

Tras haber creado el grupo de hilos, escogeremos las tareas que van a desempeñar los usuarios creados. Para ello añadiremos los “*Valores por Defecto para Petición HTTP*” haciendo click derecho sobre nuestro grupo de hilos y seleccionando *Añadir/Elemento de Configuración/Valores por Defecto para Petición HTTP*. En esta ventana, introduciremos la IP de la máquina sobre la que vamos a hacer las peticiones, en nuestro caso 192.168.56.103, correspondiente a Ubuntu Server.

Figura 4.2: Escribiendo la IP de la máquina sobre la que se realizará el test.

A continuación escogeremos la ruta de la máquina Ubuntu Server sobre la que vamos a hacer las peticiones. Añadiremos un nuevo campo haciendo click derecho sobre el grupo de hilos y seleccionando *Añadir/Muestreador/Petición HTTP*. Le daremos el nombre de *Home Page*, ya que las peticiones las queremos hacer sobre la página principal de Apache, e introduciremos en el campo “Ruta:” el directorio raíz (/). No será necesario introducir la dirección IP de la máquina, pues ya la hemos especificado en el campo “*Valores por Defecto para Petición HTTP*”.

Figura 4.3: Ruta de la máquina sobre la que realizaremos las peticiones HTTP.

Finalmente añadiremos una gráfica sobre la que mostrar los datos haciendo click derecho

sobre el grupo de hilos y seleccionando *Añadir/Receptor/Gráfico de Resultados*. Dentro de este apartado podremos elegir entre únicamente mostrar los resultados o mostrarlos y guardarlos en un archivo; escogeremos guardarlos en el archivo **resultado.jmx**. Además haremos que se nos muestre un resumen de los datos obtenidos seleccionándolo en *Añadir/Receptor/Reporte resumen*.

Ya está todo listo para ejecutar, así que pinchamos en el icono de “play” para arrancar el test, y obtendremos la siguiente gráfica. [4]

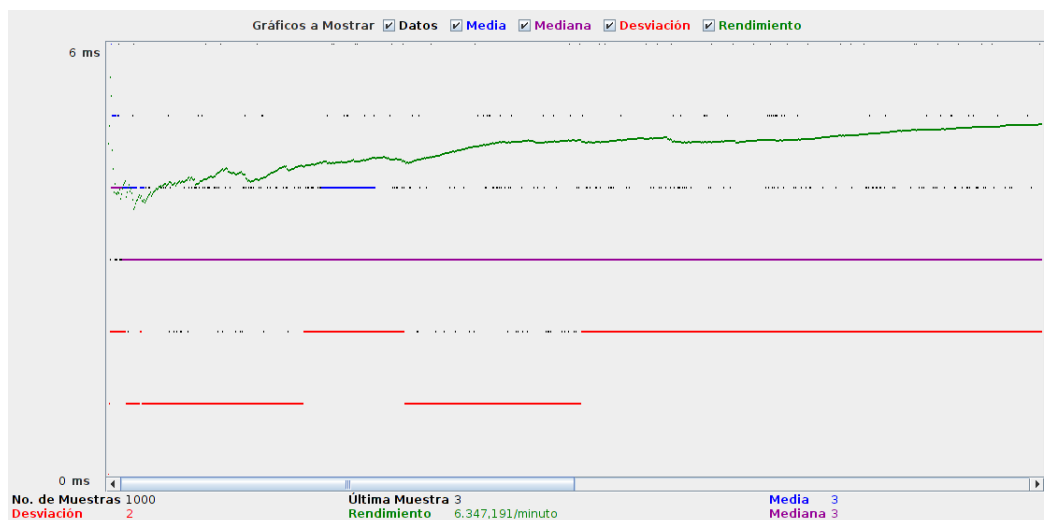


Figura 4.4: Gráfica con los resultados del test.

Podemos ver como el rendimiento va aumentando conforme avanza el test. Aún así, estos datos no son demasiado útiles para comparar el test con el conseguido con **ab**, por lo que usaremos la tabla de datos obtenida en “*Reporte resumen*”.

Reporte resumen

Nombre: Reporte resumen

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo Navegar... Log/Mostrar sólo: ☐ Escribir en Log Sólo Errores ☐ Éxitos

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Media de Bytes
Petición HTTP	1000	3	2	49	2,48	0,00%	81,8/sec	9488,53	118732,6
Total	1000	3	2	49	2,48	0,00%	81,8/sec	9488,53	118732,6

Figura 4.5: Reporte del resumen generado tras el test..

Como podemos ver en la imagen 4.5, los resultados obtenidos serán menores a los de **ab** (imagen 3.1). Con **ab** teníamos un rendimiento de 820.13 peticiones por segundo, mientras que en JMeter es de 81.8 peticiones por segundo; de la misma manera, la velocidad de transferencia con **ab** es de 95066.04 Kbytes/segundo y la de JMeter de 9488.53 Kbytes/segundo. Habrá que notar que la media de bytes transferidos de JMeter es 118732.6 Bytes, y realizando los cálculos para **ab**, que ha transferido 118698000 Bytes en 1000

peticiones, entonces la media de bytes transferidos en cada petición habrá sido de 118698 Bytes, un dato muy similar al de JMeter.

Los datos obtenidos no son los mismos que los de `ab`, pero son proporcionales: los resultados de `ab` son 10 veces mayores que los de JMeter. Esto es debido a que en `ab` se realizaban 10 peticiones cada vez y en JMeter se realizan las peticiones de una en una, por lo que el número de peticiones por segundo en `ab` será 10 veces mayor que en JMeter, así como la cantidad de Kbytes transferidos por segundo.

5. Programe un benchmark usando el lenguaje que desee o busque uno ya programado (github, openbenchmark, etc.). Debe especificar:

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

Para la realización de esta cuestión, hemos descargado el benchmark `collectl` [6] de la web proporcionada por el guión de prácticas. [5]

Objetivo del benchmark:

`Collectl` es una herramienta para monitorizar el sistema, con la diferencia a los distintos benchmarks vistos en esta práctica de que obtendremos información sobre la CPU, el disco, la red, etc. en tiempo real, pudiendo así saber qué está pasando en nuestra máquina en este momento. [6]

Métricas:

Para esta cuestión monitorizaremos el uso de la CPU y de la red de Internet; a continuación definimos las métricas de cada una de las dos: [7]

■ Monitorización de la CPU:

- `cpu`: define el porcentaje de tiempo que ha estado ocupada la CPU durante el intervalo de tiempo de monitorización. Notaremos que este porcentaje no incluye el tiempo de espera durante las operaciones de E/S.
- `sys`: porcentaje de tiempo que la CPU ha pasado en “modo sistema”, es decir, sin tener en cuenta los usuarios ni la programación de prioridades.
- `inter`: número de interrupciones por segundo.
- `ctxsw`: número de cambios de contexto por segundo.

■ Monitorización de la red:

- `KBIn`: cantidad de KB recibidos por segundo en todas las interfaces de red reales.
- `PktIn`: cantidad de paquetes recibidos por segundo todas las interfaces de red reales.

- **Size (In):** media del tamaño de los paquetes recibidos en bytes.
- **KBOut:** cantidad de KB enviados por segundo en todas las interfaces de red reales.
- **PktOut:** cantidad de paquetes enviados por segundo en todas las interfaces de red reales
- **Size (Out):** media del tamaño de los paquetes enviados en bytes.

Instrucciones para su uso:

Si ejecutamos el comando `collectl` sin argumentos, obtendremos la monitorización de la CPU, el disco y la red (aunque la red se mostrará con menos datos a los definidos en el apartado *Métricas*).

Para monitorizar únicamente la CPU, usaremos el comando `collectl -sc`, obteniendo solo los datos definidos anteriormente. [8]

Por otra parte, obtendremos los resultados de la monitorización de la red ejecutando `collectl -sn -iosize`, donde `-iosize` nos dará la opción de visualizar la media de los tamaños de los paquetes enviados y recibidos.

Ejemplo de uso:

Con la monitorización de la CPU, obtendremos los siguientes resultados.

```
simon@simon-VirtualBox:~$ collectl -sc
waiting for 1 second sample...
#<-----CPU----->
#cpu sys inter  ctxsw
 25  13   419   1407
 22   7   284   1080
  9   4   129    620
 19  11   190    913
 11   3   136    734
  7   2   116    662
  7   3   105    545
 21  11   303   1641
  9   4   105    557
 12   4   149    884
```

Figura 5.1: Monitorización de la CPU con el benchmark `collectl`.

Como podemos ver, la CPU no invierte un porcentaje de tiempo alto durante la monitorización; esto es debido a que no estamos ejecutando ningún otro proceso a la vez que el benchmark, y el porcentaje de tiempo que aparece es el que generan los procesos en segundo plano.

Para la monitorización de la red, comenzaremos sin realizar ninguna acción en la web, por lo que aparecerán todos los campos a 0, y tras ello abriremos distintos vídeos (que envían muchos datos) y realizaremos distintas búsquedas, para finalmente cerrar el navegador, momento en el que todos los campos volverán a 0.

waiting for 1 second sample...

#	KBIn	PktIn	Size	KBOut	PktOut	Size
0	3	60	0	3	68	
0	2	60	0	3	127	
9	40	218	4	32	125	
111	106	1071	5	55	94	
61	73	857	5	44	113	
69	67	1052	4	37	98	
37	75	502	8	72	107	
18	65	277	6	57	101	
41	36	1176	1	12	66	
39	93	428	11	93	121	
327	334	1002	14	94	148	
133	161	846	12	102	117	
1	8	107	1	7	160	
0	1	60	0	1	60	
30	48	644	4	34	122	
64	109	604	9	80	121	
413	341	1241	14	109	130	
207	217	977	15	88	173	

→

#	KBIn	PktIn	Size	KBOut	PktOut	Size
142	115	1263	5	38	122	
898	652	1410	7	49	143	
164	164	1025	12	62	207	
461	360	1311	7	63	106	
2060	1486	1420	7	81	95	
13	37	361	5	35	136	
11	16	700	2	10	156	
354	264	1371	4	31	128	
307	254	1237	18	104	180	
535	418	1311	11	69	168	
1477	1070	1414	10	92	108	
383	286	1371	6	33	174	
2079	1477	1441	8	102	82	
54	58	945	8	23	349	
328	265	1266	11	49	227	
0	5	60	0	5	68	
0	0	0	0	0	0	
0	2	60	0	2	60	

Figura 5.2: Monitorización de la red de Internet con el benchmark collectl.

- 5.1. Tenga en cuenta que puede comparar varios gestores de BD, lenguajes de programación web (tiempos de ejecución, gestión de memoria, ...), duración de la batería, servidor DNS, etc. Haga tres ejecuciones del mismo, cambie al menos uno de los parámetros de la máquina virtual y vuelva a ejecutar ¿es, estadísticamente hablando, significativa la diferencia entre una configuración y otra?

Para poder comparar los datos, volveremos a realizar la monitorización de la CPU, pero esta vez lanzando un ejecutable para que el porcentaje de tiempos obtenidos no se base solo en los procesos ejecutados en segundo plano.

#cpu	sys	inter	ctxsw
15	6	193	786
9	3	134	733
30	11	351	2673
17	6	209	2176
10	4	145	951
14	6	171	979
25	10	314	2020

1

#cpu	sys	inter	ctxsw
39	9	328	2847
81	11	492	3973
68	6	408	2521
67	5	408	1982
64	6	391	2204
61	4	354	1589
62	3	378	1731
56	7	354	2039
9	3	115	567
17	6	203	1562

2

#cpu	sys	inter	ctxsw
20	5	210	1087
11	4	155	894
51	11	392	3518
81	10	494	4141
65	6	554	2294
67	6	394	2231
67	6	405	2037
61	3	361	1460
65	6	404	2153
43	5	304	1591
10	4	120	598

3

#cpu	sys	inter	ctxsw
10	3	126	895
17	5	185	1175
80	12	625	4772
71	7	447	2980
67	7	414	2618
65	6	391	2224
66	5	390	1929
63	4	361	1537
65	5	407	1928
39	14	421	4129

Figura 5.3: Monitorización de la CPU lanzado procesos entre medias.

Como vemos en la imagen 5.3, cada vez que lanzamos el proceso el porcentaje de tiempo que pasa ocupada la CPU aumenta, aunque sin llegar ésta a estar completamente cargada, y al finalizar el proceso este porcentaje disminuye.

El parámetro que cambiaremos para comparar la diferencia será el número de CPUs de la máquina virtual, para así comprobar si se llega a saturar completamente. Entraremos a la configuración de la máquina virtual y reduciremos las CPUs de 2 a 1.

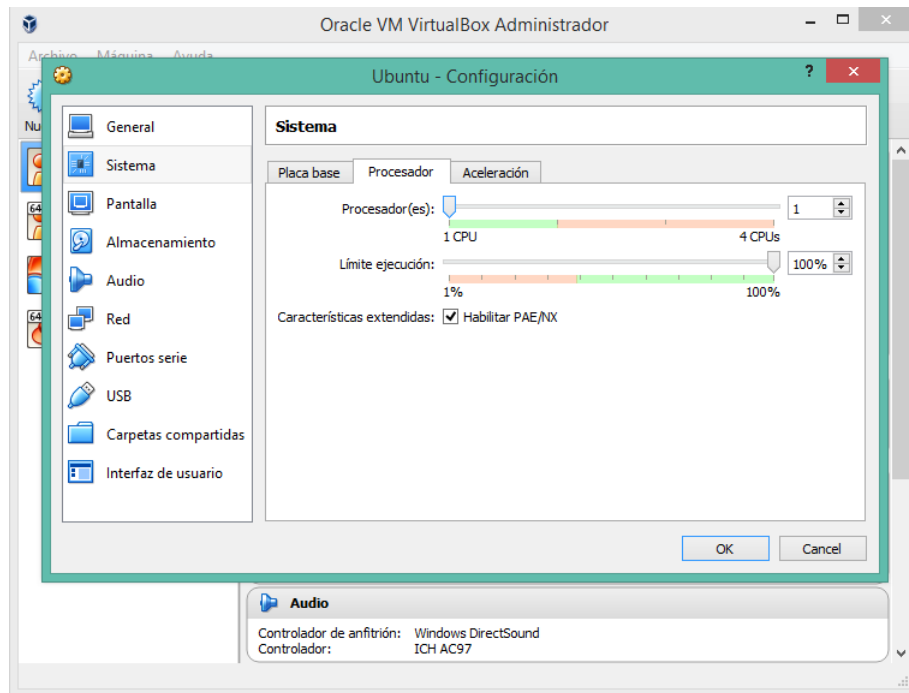


Figura 5.4: Reducción del número de CPUs de la máquina virtual.

A continuación volveremos a lanzar el benchmark y ejecutaremos el programa que sobrecargará la CPU.

```
waiting for 1 second sample...
```

#cpu	sys	inter	ctxsw
29	5	178	2109
69	14	249	4405
19	4	111	898
20	5	97	680
30	6	145	1556
100	8	341	3881
100	6	320	3100
100	2	322	2014
100	4	312	1664
100	2	297	1872
100	3	321	1707
100	2	313	1396
100	1	316	1664
38	6	150	1388
13	2	79	523

#cpu	sys	inter	ctxsw
16	2	97	811
15	3	91	635
19	3	97	721
89	10	313	4580
100	4	334	2961
100	3	337	2482
100	5	312	2367
100	2	317	1577
100	3	311	1630
100	4	315	1833
100	5	323	1906
87	2	276	1650
24	5	101	949
22	3	111	1336
62	13	288	3944

Figura 5.5: Monitorización con una única CPU.

Como podemos ver, el porcentaje de tiempo que pasa en espera únicamente con los procesos en segundo plano ya es mayor que en el caso de que tengamos 2 CPUs. Además, al ejecutar el programa que sobrecarga el procesador, el porcentaje llega al 100 %, signifi-

cando que la CPU está totalmente sobrecargada.

Por tanto, podremos afirmar que la diferencia entre las dos configuraciones es significativa.

Referencias

- [1] AB(1) - LINUX MAN PAGE, <https://linux.die.net/man/1/ab>, consultado el 10 de mayo de 2017.
- [2] PS, http://linuxcommand.org/man_pages/ps1.html, consultado el 10 de mayo de 2017.
- [3] GREP(1) - LINUX MAN PAGE, <https://linux.die.net/man/1/grep>, consultado el 10 de mayo de 2017.
- [4] 4. BUILDING A WEB TEST PLAN, <http://jmeter.apache.org/usermanual/build-web-test-plan.html>, consultado el 13 de mayo de 2017.
- [5] SOURCEFORGE, <https://sourceforge.net/directory/os:linux/?q=benchmark+cpu>, consultado el 13 de mayo de 2017.
- [6] COLLECTL, <http://collectl.sourceforge.net/>, consultado el 13 de mayo de 2017.
- [7] BRIEF DATA, <http://collectl.sourceforge.net/Data-brief.html>, consultado el 13 de mayo de 2017.
- [8] CPU MONITORING, <http://collectl.sourceforge.net/CPUs.html>, consultado el 13 de mayo de 2017.
- [9] NETWORK MONITORING, <http://collectl.sourceforge.net/Network.html>, consultado el 13 de mayo de 2017.