

Infraestructura para Big Data

Práctica 4 - Virtualización de Infraestructura

Susana Morillas Fresno

José Manuel Bustos Muñoz

Santiago Martinez De La Riva

1. **Ejercicio 0:** A. Mida el ancho de banda entre dos máquinas virtuales (vmware) con iperf durante 5 minutos. Visualice la serie temporal del ancho de banda con una muestra cada 10 segundos. ¿Es el caudal alto? ¿es estable? Recuerde estas cifras para cuando la comparemos con entornos geodistribuidos en la nube. B. Intente hacer ping a www.harvard.edu, sydney.edu.au, www.unican.es, ¿ha tenido éxito en los laboratorios? En cualquier caso, pruebe con hping3 y pinte una serie temporal con una muestra de RTT cada 10 segundos durante 5 minutos para cada sitio web. C. Periodifique las medidas de hping3 mediante cron. Cree un script que ejecute 1 iteración de hping3 (-c 1), y llámelo cada minuto con cron. ¿Ha encontrado diferencias con lo anterior?. D. Lance una máquina virtual con poco capacidad, y evalúe su capacidad con sysbench.

Medimos el ancho de banda con iperf desde una máquina virtual a una máquina ubicada en Australia, con envío udp, durante 5 minutos e intervalos de 10 segundos:

```
bigdata@bigdata:~$ iperf -c 14.1.33.133 -u -b 150m -i 10 -t 300 -r -p 5201
-----
Client connecting to 14.1.33.133, UDP port 5201
Sending 1470 byte datagrams
UDP buffer size: 208 KByte (default)
-----
[ 3] local 10.0.2.15 port 40348 connected with 14.1.33.133 port 5201
-----
Server listening on UDP port 5201
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
-----
[ ID] Interval           Transfer     Bandwidth
[ 3] 0.0-10.0 sec      179 MBytes  150 Mbits/sec
[ 3] 10.0-20.0 sec     179 MBytes  150 Mbits/sec
[ 3] 20.0-30.0 sec     179 MBytes  150 Mbits/sec
[ 3] 30.0-40.0 sec     179 MBytes  150 Mbits/sec
[ 3] 40.0-50.0 sec     179 MBytes  150 Mbits/sec
[ 3] 50.0-60.0 sec     179 MBytes  150 Mbits/sec
[ 3] 60.0-70.0 sec     160 MBytes  134 Mbits/sec
[ 3] 70.0-80.0 sec     13.5 MBytes  11.3 Mbits/sec
[ 3] 80.0-90.0 sec     41.9 MBytes  35.2 Mbits/sec
[ 3] 90.0-100.0 sec    175 MBytes  147 Mbits/sec
```

```
[ 3] 120.0-130.0 sec   9.47 MBytes  7.95 Mbits/sec
[ 3] 130.0-140.0 sec  110 MBytes  92.3 Mbits/sec
^[[ 3] 140.0-150.0 sec  178 MBytes  149 Mbits/sec
[ 3] 150.0-160.0 sec  175 MBytes  147 Mbits/sec
[ 3] 160.0-170.0 sec  58.4 MBytes  49.0 Mbits/sec
[ 3] 170.0-180.0 sec  18.5 MBytes  15.6 Mbits/sec
[ 3] 180.0-190.0 sec  149 MBytes  125 Mbits/sec
[ 3] 190.0-200.0 sec  178 MBytes  149 Mbits/sec
[ 3] 200.0-210.0 sec  159 MBytes  134 Mbits/sec
[ 3] 210.0-220.0 sec  29.1 MBytes  24.4 Mbits/sec
[ 3] 220.0-230.0 sec  54.0 MBytes  45.3 Mbits/sec
[ 3] 230.0-240.0 sec  177 MBytes  149 Mbits/sec
[ 3] 240.0-250.0 sec  179 MBytes  150 Mbits/sec
[ 3] 250.0-260.0 sec  155 MBytes  130 Mbits/sec
[ 3] 260.0-270.0 sec  19.9 MBytes  16.7 Mbits/sec
[ 3] 270.0-280.0 sec  22.4 MBytes  18.8 Mbits/sec
[ 3] 280.0-290.0 sec  166 MBytes  139 Mbits/sec
```

Al finalizar se ve el número de paquetes, y la transmisión total, con la media de Mbits por segundo.

Se obtiene un buen ancho de banda, pero se ve cierta inestabilidad con intervalos donde baja muchísimo y hay picos muy diferenciados.

```
[ 3] 290.0-300.0 sec  179 MBytes  150 Mbits/sec  
[ 3]  0.0-300.0 sec  3.70 GBytes  106 Mbits/sec  
[ 3] Sent 2700713 datagrams
```

Hacemos ping contra las 3 urls indicadas:

Ping a www.harvard.edu:

```
bigdata@bigdata:~$ ping www.harvard.edu  
PING fel.edge.pantheon.io (23.185.0.1) 56(84) bytes of data.  
64 bytes from 23.185.0.1: icmp_seq=1 ttl=63 time=12.1 ms  
64 bytes from 23.185.0.1: icmp_seq=2 ttl=63 time=11.5 ms  
64 bytes from 23.185.0.1: icmp_seq=3 ttl=63 time=11.3 ms  
64 bytes from 23.185.0.1: icmp_seq=4 ttl=63 time=11.1 ms  
64 bytes from 23.185.0.1: icmp_seq=5 ttl=63 time=11.2 ms
```

Ping a www.sydney.edu.au:

```
bigdata@bigdata:~$ ping www.sydney.edu.au  
PING www.sydney.edu.au (129.78.5.11) 56(84) bytes of data.  
64 bytes from theseymourcentre.com.au (129.78.5.11): icmp_seq=1 ttl=63 time=4  
16 ms  
64 bytes from theseymourcentre.com.au (129.78.5.11): icmp_seq=2 ttl=63 time=3  
64 ms  
64 bytes from theseymourcentre.com.au (129.78.5.11): icmp_seq=3 ttl=63 time=4  
16 ms  
64 bytes from theseymourcentre.com.au (129.78.5.11): icmp_seq=4 ttl=63 time=3  
40 ms
```

Ping a www.unican.es:

```
bigdata@bigdata:~$ ping www.unican.es  
PING siv034.unican.es (193.144.193.60) 56(84) bytes of data.  
64 bytes from SIV034.unican.es (193.144.193.60): icmp_seq=1 ttl=63 time=18.7  
ms  
64 bytes from SIV034.unican.es (193.144.193.60): icmp_seq=2 ttl=63 time=81.4  
ms  
64 bytes from SIV034.unican.es (193.144.193.60): icmp_seq=3 ttl=63 time=23.8  
ms  
64 bytes from SIV034.unican.es (193.144.193.60): icmp_seq=4 ttl=63 time=23.5  
ms  
64 bytes from SIV034.unican.es (193.144.193.60): icmp_seq=5 ttl=63 time=27.2  
ms  
64 bytes from SIV034.unican.es (193.144.193.60): icmp_seq=6 ttl=63 time=27.0  
ms
```

Ahora usamos hping3 con las 3 urls durante mayor tiempo con intervalos de 10 segundos, para obtener unos datos más fiables:

Hping3 a www.harvard.edu:

```
bigdata@bigdata:~$ sudo hping3 -S -p 80 www.harvard.edu -i 10 -t 300
HPING www.harvard.edu (eth0 23.185.0.1): S set, 40 headers + 0 data bytes
len=46 ip=23.185.0.1 ttl=64 id=2236 sport=80 flags=SA seq=0 win=65535 rtt=12.6 ms
len=46 ip=23.185.0.1 ttl=64 id=2237 sport=80 flags=SA seq=1 win=65535 rtt=11.2 ms
len=46 ip=23.185.0.1 ttl=64 id=2238 sport=80 flags=SA seq=2 win=65535 rtt=6.7 ms
len=46 ip=23.185.0.1 ttl=64 id=2239 sport=80 flags=SA seq=3 win=65535 rtt=10.7 ms
...
--- www.harvard.edu hping statistic ---
31 packets transmitted, 31 packets received, 0% packet loss
round-trip min/avg/max = 6.7/15.7/112.9 ms
```

Hping3 a www.sydney.edu.au:

```
bigdata@bigdata:~$ sudo hping3 -S -p 80 www.sydney.edu.au -i 10 -t 300
HPING www.sydney.edu.au (eth0 129.78.5.11): S set, 40 headers + 0 data bytes
len=46 ip=129.78.5.11 ttl=64 id=2268 sport=80 flags=SA seq=0 win=65535 rtt=36.8 ms
len=46 ip=129.78.5.11 ttl=64 id=2269 sport=80 flags=SA seq=1 win=65535 rtt=11.7 ms
len=46 ip=129.78.5.11 ttl=64 id=2270 sport=80 flags=SA seq=2 win=65535 rtt=15.4 ms
len=46 ip=129.78.5.11 ttl=64 id=2271 sport=80 flags=SA seq=3 win=65535 rtt=10.8 ms
...
--- www.sydney.edu.au hping statistic ---
31 packets transmitted, 31 packets received, 0% packet loss
round-trip min/avg/max = 4.1/16.5/151.4 ms
```

Hping3 a www.unican.es:

```
bigdata@bigdata:~$ sudo hping3 -S -p 80 www.unican.es -i 10 -t 300
HPING www.unican.es (eth0 193.144.193.60): S set, 40 headers + 0 data bytes
len=46 ip=193.144.193.60 ttl=64 id=2300 sport=80 flags=SA seq=0 win=65535 rtt=5.6 ms
len=46 ip=193.144.193.60 ttl=64 id=2301 sport=80 flags=SA seq=1 win=65535 rtt=17.3 ms
len=46 ip=193.144.193.60 ttl=64 id=2302 sport=80 flags=SA seq=2 win=65535 rtt=45.1 ms
len=46 ip=193.144.193.60 ttl=64 id=2303 sport=80 flags=SA seq=3 win=65535 rtt=12.4 ms
len=46 ip=193.144.193.60 ttl=64 id=2304 sport=80 flags=SA seq=4 win=65535 rtt=...
...
--- www.unican.es hping statistic ---
31 packets transmitted, 31 packets received, 0% packet loss
round-trip min/avg/max = 3.6/14.2/45.1 ms
```

Con hping3 y durante mayor tiempo se obtienen unos datos más fiables, y observamos como la latencia suele acabar convergiendo bastante.

Programamos un script para que se ejecute cada minuto, y haga un hping3 a una url:

Se crea un script para lanzar el comando:

```
sudo hping3 -c 1 -S -p 80 www.unican.es > ./fichero.txt
```

En el cron se indica que se ejecute:

```
* * * * * /home/bigdata/script.sh
```

Evaluar capacidad con sysbench:

Sacamos los datos a un fichero y lo visualizamos después:

```
bigdata@bigdata:~$ sysbench --test=cpu --cpu-max-prime=5000 --num-threads=2 run  
>> resultados2.out
```

```
GNU nano 2.2.6      Archivo: resultados2.out  
  
Maximum prime number checked in CPU test: 5000  
  
Test execution summary:  
  total time:                2.9905s  
  total number of events:     10000  
  total time taken by event execution: 5.9785  
  per-request statistics:  
    min:                     0.29ms  
    avg:                     0.60ms  
    max:                     20.36ms  
    approx. 95 percentile:    0.37ms  
  
Threads fairness:  
  events (avg/stddev):       5000.0000/2.00  
  execution time (avg/stddev): 2.9893/0.00
```

Ahora volvemos a lanzar sysbench en la máquina virtual, pero hemos subido la RAM de la máquina de 2GB a 4GB, y hemos subido de 1 cpu a 2 cpus.

Obtenemos los siguientes resultados:

```
bigdata@bigdata:~$ sysbench --test=cpu --cpu-max-prime=5000 --num-threads=2 run
>> resultados3.out
bigdata@bigdata:~$ nano resultados3.out
```

```
GNU nano 2.2.6      Archivo: resultados3.out

Maximum prime number checked in CPU test: 5000

Test execution summary:
  total time:                1.5212s
  total number of events:    10000
  total time taken by event execution: 3.0405
  per-request statistics:
    min:                     0.28ms
    avg:                     0.30ms
    max:                     1.53ms
    approx. 95 percentile:   0.36ms

Threads fairness:
  events (avg/stddev):       5000.0000/1.00
  execution time (avg/stddev): 1.5203/0.00
```

Parece obtenerse una mejoría en el rendimiento o velocidad.

Uso de fichero testcpu.sh:

Enviamos a la máquina virtual el fichero testcpu.sh, y lo lanzamos enviando la salida a un fichero .log:

```
bigdata@bigdata:~$ ./testcpu.sh > /home/bigdata/salida.log
```

Lanzamos entonces tanto sysbench, como /proc/stat:

```
[1] + Detenido                               nano resultados4.out
bigdata@bigdata:~$ sysbench --test=cpu --cpu-max-prime=5000 --num-threads=2 run
>> resultados.out
bigdata@bigdata:~$ cat /proc/stat | grep '^cpu' >> resultados2.out
bigdata@bigdata:~$
```

Luego paramos el script testcpu.sh y vemos la salida obtenida, donde puede verse algunos picos de uso que deberían corresponder con las herramientas lanzadas anteriormente:

```
CPU: 0%
CPU: 0%
CPU: 0%
CPU: 0%
CPU: 1%
CPU: 0%
CPU: 2%
CPU: 1%
CPU: 47%
CPU: 100%
CPU: 8%
CPU: 0%
CPU: 1%
CPU: 1%
CPU: 0%
CPU: 0%
CPU: 1%
CPU: 0%
CPU: 0%
```

2. Ejercicio 1: Mida la latencia entre las VMs lanzadas anteriormente, Europa norte y occidental, Sudeste asiático, Sudamérica y Australia. ¿En cuáles se podría desplegar un servicio de VoIP en España?

Al estar caídas a la hora de realizar la práctica las máquinas usadas en clase, podemos utilizar la web "www.cloudping.info" para hacer ping contra diversas máquinas repartidas por el mundo y poder ver la latencia y poder decidir cuales serían válidas para desplegar un servicio VoIP desde España.

Los resultados que se obtienen son:

Region	Latency
US-East (Virginia)	93 ms
US East (Ohio)	110 ms
US-West (California)	189 ms
US-West (Oregon)	212 ms
Canada (Central)	109 ms
Europe (Ireland)	39 ms
Europe (London)	30 ms
Europe (Frankfurt)	38 ms
Europe (Paris)	28 ms
Europe (Stockholm)	52 ms
Asia Pacific (Mumbai)	163 ms
Asia Pacific (Osaka-Local)	300 ms
Asia Pacific (Seoul)	318 ms
Asia Pacific (Singapore)	309 ms
Asia Pacific (Sydney)	402 ms
Asia Pacific (Tokyo)	301 ms
South America (São Paulo)	241 ms
China (Beijing)	264 ms
China (Ningxia)	266 ms
AWS GovCloud (US-East)	108 ms
AWS GovCloud (US)	217 ms

HTTP Ping

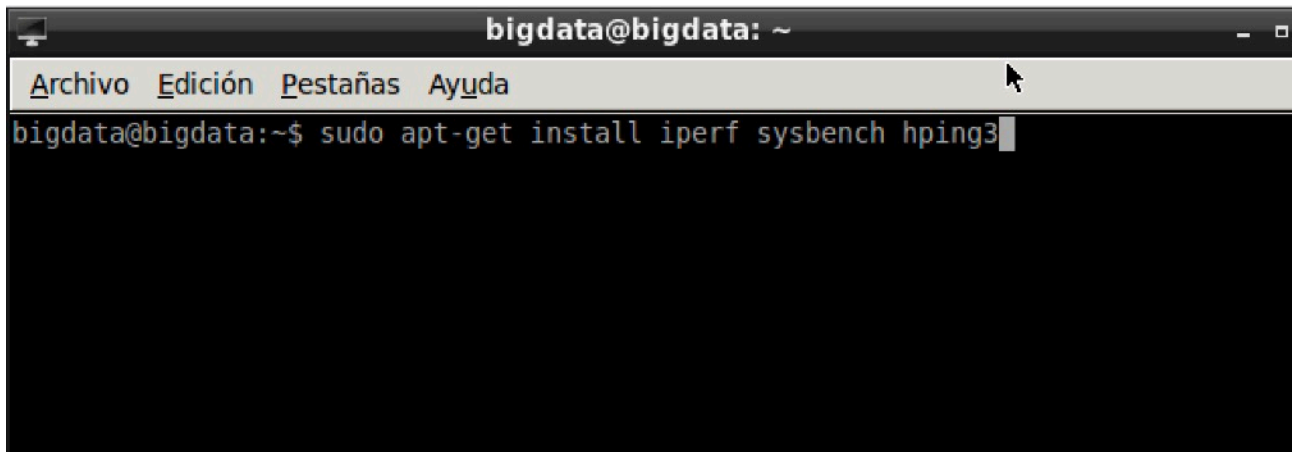
En general para tener una buena calidad de servicio no se deben superar los 150 ms de latencia, por lo tanto viendo los resultados podemos decir que las áreas en las que se podría desplegar un servicio VoIP en España serían: Europa y la costa Este de EEUU y de Canada.

También podemos usar las máquinas proporcionadas en la web "<https://iperf.cc/es/>" y por ejemplo usando hping3 desde una máquina virtual Ubuntu comprobar o verificar lo dicho anteriormente.

Usamos 4 máquinas en distintas áreas: Francia, USA, Yakarta y Australia:

- Francia: IP → 89.84.1.222, url → bouygues.testdebit.info
- USA: IP → 209.132.160.146, url → speedtest.weendeavor.com
- Yakarta: IP → 117.102.109.186, url → iperf.biznetnetworks.com
- Australia: IP → 14.1.33.133, url → iperf-akl.as45177.net

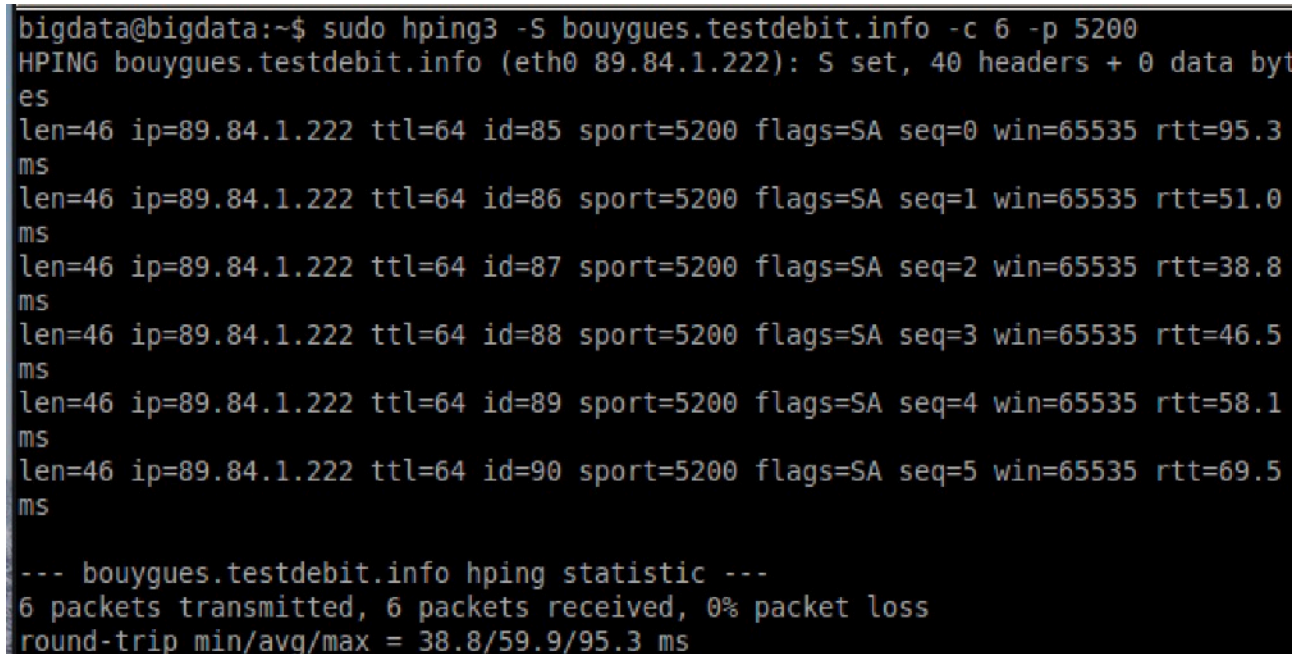
Primero instalamos en la máquina virtual las herramientas necesarias:

A terminal window titled 'bigdata@bigdata: ~' with a menu bar containing 'Archivo', 'Edición', 'Pestañas', and 'Ayuda'. The command 'sudo apt-get install iperf sysbench hping3' is entered and executed, with a cursor at the end of the line.

```
bigdata@bigdata:~$ sudo apt-get install iperf sysbench hping3
```

Lanzamos las pruebas con hping3:

Máquina de Paris:

A terminal window showing the output of the command 'sudo hping3 -S bouygues.testdebit.info -c 6 -p 5200'. It displays six individual packet test results with details like IP, TTL, ID, sport, flags, seq, win, and rtt. It concludes with a summary: '--- bouygues.testdebit.info hping statistic ---', '6 packets transmitted, 6 packets received, 0% packet loss', and 'round-trip min/avg/max = 38.8/59.9/95.3 ms'.

```
bigdata@bigdata:~$ sudo hping3 -S bouygues.testdebit.info -c 6 -p 5200
HPING bouygues.testdebit.info (eth0 89.84.1.222): S set, 40 headers + 0 data bytes
len=46 ip=89.84.1.222 ttl=64 id=85 sport=5200 flags=SA seq=0 win=65535 rtt=95.3 ms
len=46 ip=89.84.1.222 ttl=64 id=86 sport=5200 flags=SA seq=1 win=65535 rtt=51.0 ms
len=46 ip=89.84.1.222 ttl=64 id=87 sport=5200 flags=SA seq=2 win=65535 rtt=38.8 ms
len=46 ip=89.84.1.222 ttl=64 id=88 sport=5200 flags=SA seq=3 win=65535 rtt=46.5 ms
len=46 ip=89.84.1.222 ttl=64 id=89 sport=5200 flags=SA seq=4 win=65535 rtt=58.1 ms
len=46 ip=89.84.1.222 ttl=64 id=90 sport=5200 flags=SA seq=5 win=65535 rtt=69.5 ms
--- bouygues.testdebit.info hping statistic ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 38.8/59.9/95.3 ms
```

Máquina de USA:

```
--- 209.132.160.146 hping statistic ---  
6 packets transmitted, 6 packets received, 0% packet loss  
round-trip min/avg/max = 143.1/161.6/178.6 ms
```

Máquina de Yakarta:

```
--- 117.102.109.186 hping statistic ---  
6 packets transmitted, 6 packets received, 0% packet loss  
round-trip min/avg/max = 290.4/341.3/397.9 ms
```

Máquina de Auckland:

```
--- 14.1.33.133 hping statistic ---  
6 packets transmitted, 6 packets received, 0% packet loss  
round-trip min/avg/max = 368.4/433.9/569.9 ms
```

Con estas máquinas en las que se ha estudiado la latencia mediante hping3 sólo la de Francia (Europa) serviría para un servicio VoIP en España. La de USA supera en algo el límite considerado como aceptable para el servicio, seguramente porque estará ubicada al menos en la parte central de EEUU.

3. Ejercicio 2: Mida el ancho de banda entre las VMs lanzadas anteriormente, Europa norte y occidental, Sudeste asiático, Sudamérica y Australia. Describa los resultados y compárelos con los resultados del ejercicio 0.

Desde la máquina virtual lanzamos Iperf a las distintas máquinas para ver los resultados:

Máquina de Francia:

```
bigdata@bigdata:~$ iperf -c 89.84.1.222 -m -i 5 -t 30 -r -p 5200
bind failed: Address already in use
-----
Client connecting to 89.84.1.222, TCP port 5200
TCP window size: 128 KByte (default)
-----
[ 3] local 10.0.2.15 port 40900 connected with 89.84.1.222 port 5200
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0- 5.0 sec  90.6 KBytes  148 Kbits/sec
[ 3] 5.0-10.0 sec  0.00 0.00s 29514790517934759936 Bytes/sec
[ 3] 10.0-15.0 sec  0.00 0.00s 29514790517934755840 Bytes/sec
[ 3] 15.0-20.0 sec  0.00 0.00s 29514790517934759936 Bytes/sec
[ 3] 20.0-25.0 sec  0.00 0.00s 29514790517934755840 Bytes/sec
[ 3] 25.0-30.0 sec  0.00 0.00s 29514790517934755840 Bytes/sec
[ 3] 0.0-30.0 sec  0.00 0.00s 4918856624941579264 Bytes/sec
[ 3] MSS size 1460 bytes (MTU 1500 bytes, ethernet)
bigdata@bigdata:~$
```

También podemos lanzar iperf y almacenar los resultados en un fichero y verlo en consola:

```
iperf -c 89.84.1.222 -p 5201 -w 100000 >> resultados.out
vim ./resultados.out
```

```
-----
Client connecting to 89.84.1.222, TCP port 5200
TCP window size: 230 KByte (default)
-----
[ 3] local 10.0.2.15 port 40902 connected with 89.84.1.222 port 5200
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0- 5.0 sec  99.1 KBytes  162 Kbits/sec
-----
Client connecting to 89.84.1.222, TCP port 5201
TCP window size: 195 KByte (WARNING: requested 97.7 KByte)
-----
[ 3] local 10.0.2.15 port 34242 connected with 89.84.1.222 port 5201
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-10.0 sec  0.00 <88> 0s 14751242515714086912 Bytes/sec
-----
Client connecting to 89.84.1.222, TCP port 5201
TCP window size: 195 KByte (WARNING: requested 97.7 KByte)
-----
[ 3] local 10.0.2.15 port 34244 connected with 89.84.1.222 port 5201
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-10.0 sec  0.00 <88> 0s 14747140097743476736 Bytes/sec
-----
"./resultados.out" [convertido] 28L, 1604C          1,1      Comienzo
```

Máquina de USA:

```
bigdata@bigdata:~$ iperf -c 209.132.160.146 -m -i 5 -t 30 -r -p 5201
-----
Server listening on TCP port 5201
TCP window size: 85.3 KByte (default)
-----
Client connecting to 209.132.160.146, TCP port 5201
TCP window size: 128 KByte (default)
-----
[ 3] local 10.0.2.15 port 54066 connected with 209.132.160.146 port 5201
[ ID] Interval      Transfer    Bandwidth
[ 3]  0.0- 5.0 sec   101 KBytes  165 Kbits/sec
[ 3]  5.0-10.0 sec  0.00 0 00s  29514790517934755840 Bytes/sec
[ 3] 10.0-15.0 sec  0.00 0 00s  29514790517934755840 Bytes/sec
[ 3] 15.0-20.0 sec  0.00 0 00s  29514790517934755840 Bytes/sec
[ 3] 20.0-25.0 sec  0.00 0 00s  29514790517934755840 Bytes/sec
[ 3] 25.0-30.0 sec  0.00 0 00s  29514790517934759936 Bytes/sec
[ 3]  0.0-30.0 sec  0.00 0 00s  4918510545106955264 Bytes/sec
[ 3] MSS size 1460 bytes (MTU 1500 bytes, ethernet)
```

Máquina de Yakarta:

```
bigdata@bigdata:~$ iperf -c 117.102.109.186 -m -i 5 -t 30 -r -p 5201
bind failed: Address already in use
-----
Client connecting to 117.102.109.186, TCP port 5201
TCP window size: 128 KByte (default)
-----
[ 3] local 10.0.2.15 port 40804 connected with 117.102.109.186 port 5201
[ ID] Interval      Transfer    Bandwidth
[ 3]  0.0- 5.0 sec   498 KBytes  815 Kbits/sec
[ 3]  5.0-10.0 sec  0.00 0 00s  29514790517934964736 Bytes/sec
[ 3] 10.0-15.0 sec  0.00 0 00s  29514790517934755840 Bytes/sec
[ 3] 15.0-20.0 sec  0.00 0 00s  29514790517934755840 Bytes/sec
[ 3] 20.0-25.0 sec  0.00 0 00s  29514790517934755840 Bytes/sec
[ 3] 25.0-30.0 sec  0.00 0 00s  29514790517934755840 Bytes/sec
[ 3]  0.0-30.0 sec  0.00 0 00s  4917959967060793344 Bytes/sec
[ 3] MSS size 1460 bytes (MTU 1500 bytes, ethernet)
```

Máquina de Australia:

```
bigdata@bigdata:~$ iperf -c 14.1.33.133 -m -i 5 -t 30 -r -p 5201
bind failed: Address already in use
-----
Client connecting to 14.1.33.133, TCP port 5201
TCP window size: 128 KByte (default)
-----
[ 4] local 10.0.2.15 port 51010 connected with 14.1.33.133 port 5201
[ ID] Interval      Transfer    Bandwidth
[ 4]  0.0- 5.0 sec   190 KBytes  312 Kbits/sec
[ 4]  5.0-10.0 sec  0.00 0 00s  29514790517934755840 Bytes/sec
[ 4] 10.0-15.0 sec  0.00 0 00s  29514790517934755840 Bytes/sec
[ 4] 15.0-20.0 sec  0.00 0 00s  29514790517934755840 Bytes/sec
[ 4] 20.0-25.0 sec  0.00 0 00s  29514790517934755840 Bytes/sec
[ 4] 25.0-30.0 sec  0.00 0 00s  29514790517934755840 Bytes/sec
[ 4]  0.0-30.0 sec  0.00 0 00s  4918690382505169920 Bytes/sec
[ 4] MSS size 1460 bytes (MTU 1500 bytes, ethernet)
```


Se han obtenido unos resultados bastante pobres al lanzar Iperf contra las distintas máquinas desde nuestra máquina virtual.

Vamos a probar contra una de ellas que acepta conexión udp, este otro tipo de envío. Para ello usamos la bandera -u:

```
bigdata@bigdata:~$ iperf -c 14.1.33.133 -u -m -i 5 -t 30 -r -p 5201
-----
Client connecting to 14.1.33.133, UDP port 5201
Sending 1470 byte datagrams
UDP buffer size: 208 KByte (default)
-----
[ 3] local 10.0.2.15 port 57060 connected with 14.1.33.133 port 5201
-----
Server listening on UDP port 5201
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
-----
[ ID] Interval           Transfer     Bandwidth
[ 3]  0.0- 5.0 sec      630 KBytes  1.03 Mbits/sec
[ 3]  5.0-10.0 sec     633 KBytes  1.04 Mbits/sec
[ 3] 10.0-15.0 sec     633 KBytes  1.04 Mbits/sec
[ 3] 15.0-20.0 sec     633 KBytes  1.04 Mbits/sec
[ 3] 20.0-25.0 sec     633 KBytes  1.04 Mbits/sec
[ 3] 25.0-30.0 sec     632 KBytes  1.03 Mbits/sec
[ 3]  0.0-30.0 sec    3.71 MBytes  1.04 Mbits/sec
[ 3] Sent 2676 datagrams
```

Vemos como se obtiene un mejor resultado, con 1Mbits/segundo, pero vamos a intentar mejorarlo usando "-b 150m" ya que las conexiones udp vienen limitadas por defecto a 1Mbit, y con este cambio intentamos poner otro límite de ancho de banda contra el que probar:

```
bigdata@bigdata:~$ iperf -c 14.1.33.133 -u -b 150m -i 5 -t 30 -r -p 5201
-----
Client connecting to 14.1.33.133, UDP port 5201
Sending 1470 byte datagrams
UDP buffer size: 208 KByte (default)
-----
[ 3] local 10.0.2.15 port 33989 connected with 14.1.33.133 port 5201
-----
Server listening on UDP port 5201
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
-----
[ ID] Interval           Transfer     Bandwidth
[ 3]  0.0- 5.0 sec    89.3 MBytes  150 Mbits/sec
[ 3]  5.0-10.0 sec    89.5 MBytes  150 Mbits/sec
[ 3] 10.0-15.0 sec    89.5 MBytes  150 Mbits/sec
[ 3] 15.0-20.0 sec    89.5 MBytes  150 Mbits/sec
[ 3] 20.0-25.0 sec    89.6 MBytes  150 Mbits/sec
[ 3] 25.0-30.0 sec    89.6 MBytes  150 Mbits/sec
[ 3]  0.0-30.0 sec   537 MBytes  150 Mbits/sec
[ 3] Sent 383080 datagrams
```

Así obtenemos un ancho de banda mucho mejor.

4. Ejercicio 3 (se recomienda usar una tabla tipo Excel): Calcular costes de un despliegue de 100 VMs durante 1 mes de 31 días en un centro de datos por especificar (requisitos mínimos).

Se adjunta excel "P4_Ejercicio3" con todos los cálculos realizados.

Finalmente se recomienda la opción de menor coste, que corresponde a la sombreada en amarillo en la imagen:

Con capacidad 64 x ECU (2 GHz Xeon) + 64 GB RAM durante las 24 horas del día 31																
UE (Francfort)	m5.4xlarge	16	60	64 Solo EBS	0,92	20000	0	80	0,02	10	0,02	10	0,02	100	1	11408
UE (Francfort)	m5d.4xlarge	16	60	64 2 x 300 SSD NVMe	1,088	20000	0	80	0,02	10	0,02	10	0,02	100	1	11811,2
EEUU Este (Ohio)	m5.4xlarge	16	60	64 Solo EBS	0,768	20000	0	80	0,02	10	0,02	10	0,02	100	1	10043,2
EEUU Este (Ohio)	m5d.4xlarge	16	60	64 2 x 300 SSD NVMe	0,904	20000	0	80	0,02	10	0,02	10	0,02	100	1	10369,6
América del Sur (Sao Paulo)	m5.4xlarge	16	60	64 Solo EBS	1,224	20000	0,01	80	0	10	0,16	10	0,16	100	1	20257,6
Asia Pacif (Singapur)	m5.4xlarge	16	60	64 Solo EBS	0,96	20000	0,01	80	0,09	10	0,09	10	0	100	1	13114
Asia Pacif (Singapur)	m5d.4xlarge	16	60	64 2 x 300 SSD NVMe	1,128	20000	0,01	80	0,09	10	0,09	10	0	100	1	13517,2
Asia Pacif (Sidney)	m5.4xlarge	16	60	64 Solo EBS	0,96	20000	0,01	80	0,14	10	0,14	10	0,14	100	1	13704
Asia Pacif (Sidney)	m5d.4xlarge	16	60	64 2 x 300 SSD NVMe	1,136	20000	0,01	80	0,14	10	0,14	10	0,14	100	1	14126,4

5. Ejercicio 7 (optativo). Lea: <https://www.dummies.com/programming/networking/how-to-calculate-the-cost-of-applications-in-a-cloud-computing-data-center/>. Resuma los consejos para el despliegue de un centro de datos privado.

No siempre el centro de datos en la nube será la mejor opción, hay que calcular el coste con precisión para evaluar si merece la pena en el aspecto económico el pasar del centro de datos actual a uno en la nube.

Al comparar económicamente las dos soluciones se deben tener en cuenta los siguientes aspectos, que serán costos a tener en cuenta en los componentes:

- Costos del servidor
- Costos de almacenamiento
- Costos de red
- Copia de seguridad y costos de archivo
- Costos de recuperación de desastres
- Costos de la infraestructura del centro de datos
- Costos de la plataforma
- Costos de mantenimiento del software (paquete de software)
- Costos de mantenimiento del software (software interno)
- Costos de soporte de la mesa de ayuda
- Costos del personal de soporte operativo
- Costos de software de infraestructura

Los costos deben compararse de forma justa, ya que no es lo mismo que una aplicación esté en la nube o no. Por ejemplo hay que tener en cuenta a la hora de elegir un proveedor cloud la capacidad que tiene de recuperación ante cualquier problema. También una buena opción para ver si es buena opción la nube es si tenemos poco espacio en el centro de datos actual, ya que la nube será una opción más económica para escalar.

La forma de ser más justos es calcular los costes para cada aplicación y ver el total, y además es una buena idea comparar el coste total de tener una aplicación en propiedad con el coste de ejecutarla en la nube, y ya elegir esta opción si es menor y además se cumplen otras premisas comentadas anteriormente.

Al final para tomar la decisión como hemos comentado deben compararse y tenerse en cuenta los costes totales, y ponderarlo con puntos que la nube puede darte como ventajas: ahorro en costos de energía, menor complejidad en la gestión, estabilidad del software o pruebas de diagnóstico.

6. Ejercicio 7 II sesión: Lanzar una maquina virtual en DevStack mediante la interface gráfica, medir sus capacidades (CPU con sysbench) de manera periódica con cron volcando a fichero. Visualizar la capacidad con el tiempo. Después, habiéndose dejado que se tomen una serie de medidas, redimensionar la máquina virtual (en num. cores, en RAM, etc.), y tras comprobar la corrección de la ejecución, visualizar los nuevos resultados. Discuta la salida.

Información de la instancia creada:

```
ubuntu@team69:~$ nproc
```

```
1
```

```
ubuntu@team69:~$ free -m
```

	total	used	free	shared	buff/cache	available
Mem:	968	39	712	2	216	780
Swap:	0	0	0			

Creamos un script para que se ejecute la herramienta sysbench y que se compruebe el test de la CPU solicitado:

```
ubuntu@team69:~/$ pico /home/ubuntu/scripts/cpu-sysbench7.sh
```

```
# !/bin/bash
```

```
sysbench --test=cpu --cpu-max-prime=20000 --num-threads=2 run
```

A continuación actualizamos nuestro gestor de paquetes e instalamos la herramienta:

```
ubuntu@team69:~/$ sudo apt-get update
```

```
ubuntu@team69:~/$ sudo apt-get install sysbench
```

Luego introducimos una tarea programada en el cron del sistema para que se ejecute cada 2 min y lance el script creado y guarde la salida en el fichero "cpu-sysbench7.out":

```
ubuntu@team69:~$ crontab -e
```

```
Select an editor. To change later, run 'select-editor'.
```

```
1. /bin/ed
```

```
2. /bin/nano <---- easiest
```

```
3. /usr/bin/vim.basic
```

```
4. /usr/bin/vim.tiny
```

```
Choose 1-4 [2]: 2
```

```
crontab: installing new crontab
```

```
# Edit this file to introduce tasks to be run by cron.
```

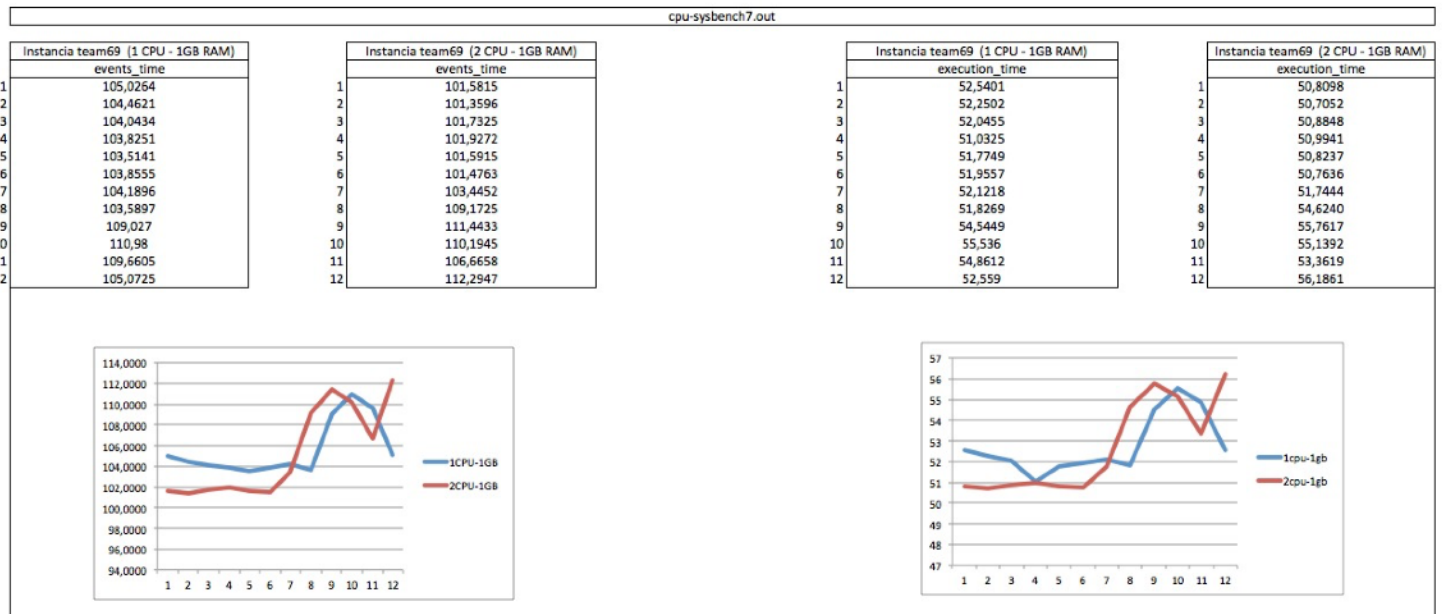
```
# ...
```

```
*/* * * * * root bash /home/ubuntu/scripts/cpu-sysbench7.sh >> /home/ubuntu/cpu-sysbench7.out
```

```
ubuntu@team69:~$ crontab -e
```

```
crontab: installing new crontab
```

Una vez ejecutado esto se genera el fichero cpu-sysbench7.out que contiene todas las trazas de la primera fase y la segunda una vez redimensionada la instancia, lo que nos permite hacer los análisis y conclusiones solicitados en el ejercicio:



Los tiempos obtenidos en la primera fase con una instancia de 1 CPU y 1GB de RAM son menores que cuando redimensionamos la instancia y le asignamos 2 CPUs. Esto no tiene en principio mucho sentido, pues si la máquina tiene más cores, la lógica nos haría pensar que tiene más capacidad de procesamiento, y que en el segundo caso la CPU debería de tardar menos tiempo en realizar los test, pero esto no ocurre así.

Hemos investigado más sobre este caso y puede ser que la instalación que hemos hecho no tenga la configuración correcta, o no soporte KVM anidados como comentas en este hilo oficial de OpenStack: <https://ask.openstack.org/en/question/94405/very-slow-cpu-for-compute-instances-on-tripleo-in-virtual-environment/>

7. **Ejercicio 8 II sesión:** Mediante comandos, suba una imagen determinada de Ubuntu, lance una máquina virtual deje un elemento de medida periódica de capacidad (e.g., sysbench con cron o alguno script iterador), deje que se tomen una serie de medidas, redimensione la máquina virtual (a 2 CPUs y 1 GB RAM, mismo disco duro), y tras comprobar la corrección de la ejecución, visualice los resultados. E.g., vuelque los datos en un fichero y cópieselo al host con scp. Incluya los comandos en la memoria (en forma de script), visualización de la capacidad de la CPU con el tiempo, y discuta los resultados.

Virtualización de Infraestructuras mediante Scripts de comandos:

- Accedemos a Openrc para poder gestionar nuestro Cloud desde la consola:

```
sudo su - stack
```

```
cd devstack
```

```
source openrc admin admin
```

- Listamos las imágenes:

```
stack@ubuntu:~/devstack$glance image-list
```

ID	Name
f99072e2-22f0-4a8e-a1a8-b9788f3d33b4	cirros-0.3.5-x86_64-disk
dbb3bb4f-c95a-4764-a811-57499ba3b9fb	ubuntu

- Creamos una nueva instancia a partir de la imagen descargada en el ejercicio 7:

```
stack@ubuntu:~/devstack$glance image-create --name ubuntuByComandos --container-format bare --disk-format qcow2 --visibility public --file iso/xenial-server-cloudimg-amd64-disk1.img
```

Property	Value
checksum	690e089e7d197f9664d5a1e723421b4d
container_format	bare
created_at	2019-03-15T19:29:45Z
disk_format	qcow2
id	8bf1ab76-c4f1-43c1-b5c8-b3feb5f64ee0
min_disk	0
min_ram	0
name	ubuntuByComandos
owner	95b1989d2c7c42daa1081a8e13d40e16
protected	False
size	289669120
status	active
tags	[]
updated_at	2019-03-15T19:29:49Z
virtual_size	None
visibility	public

- Comprobamos el listado de flavor que tenemos disponibles. Podemos observar el flavor creado en la parte de interfaz suriFlavor:

```
stack@ubuntu:~/devstack$ nova flavor-list
```

ID	Name	Memory_MB	Disk	Ephemeral	Swap	VCPUs	RXTX_Factor	Is_Public
1	m1.tiny	512	1	0	1	1.0	True	
2	m1.small	2048	20	0	1	1.0	True	
267c320f-6368-4d6b-ba6c-1a6bd65a21d7	suriFlavor	1000	3	0	1	1.0	True	
3	m1.medium	4096	40	0	2	1.0	True	
4	m1.large	8192	80	0	4	1.0	True	
42	m1.nano	64	0	0	1	1.0	True	
5	m1.xlarge	16384	160	0	8	1.0	True	
84	m1.micro	128	0	0	1	1.0	True	
c1	cirros256	256	0	0	1	1.0	True	
d1	ds512M	512	5	0	1	1.0	True	
d2	ds1G	1024	10	0	1	1.0	True	
d3	ds2G	2048	10	0	2	1.0	True	
d4	ds4G	4096	20	0	4	1.0	True	

- Creamos un nuevo flavor mediante comando llamado suriFlavorByComando:

```
stack@ubuntu:~/devstack$ nova flavor-create suriFlavorByComando dpp 1000 3 1
```

ID	Name	Memory_MB	Disk	Ephemeral	Swap	VCPUs	RXTX_Factor	Is_Public
dpp	suriFlavorByComando	1000	3	0	1	1.0	True	

- Listamos los grupos de seguridad existentes:

```
stack@ubuntu:~/devstack$ openstack security group list
```

ID	Name	Description	Project
12a753ba-e418-4944-8298-552128657d32	default	Default security group	
b7b7be1881c24beb81fcae701d33d975			
46ff4360-92fd-4701-9bcc-84088e86c3c6	default	Default security group	
95b1989d2c7c42daa1081a8e13d40e16			
84d02066-1414-418b-9ae6-782f62dadfde	default	Default security group	
a0e4fe24-21e8-40c4-b1bc-6130e649333b	default	Default security group	
5574c86c900149dc93ed069d9e8a1a62			

- Creamos un nuevo grupo de seguridad:

```
stack@ubuntu:~/devstack$ openstack security group create defaultByComando
```

```
+-----+
+-----+
+-----+
| Field      | Value
|
+-----+
+-----+
+-----+
| created_at   | 2019-03-15T19:37:36Z
|
| description  | defaultByComando
|
| id           | c0964b92-d480-4f0f-ab9d-ca69f554469b
|
| name         | defaultByComando
|
| project_id   | 95b1989d2c7c42daa1081a8e13d40e16
|
| revision_number | 2
|
| rules        | created_at='2019-03-15T19:37:36Z', direction='egress', ethertype='IPv4',
id='7e86c5e7-d825-4fc8-8c29-ed350dae9dc4', updated_at='2019-03-15T19:37:36Z' |
|              | created_at='2019-03-15T19:37:36Z', direction='egress', ethertype='IPv6',
id='c982ab98-58f4-430e-aabd-90d27bbe3230', updated_at='2019-03-15T19:37:36Z' |
| updated_at   | 2019-03-15T19:37:36Z
|
+-----+
+-----+
+-----+
```

- Creamos una nueva regla para SSH para permitir el acceso desde la red:

```
stack@ubuntu:~/devstack$ openstack security group rule create defaultByComando --protocol tcp
--dst-port 22:22 --remote-ip 0.0.0.0/0
```

```
+-----+
+-----+
+-----+
| Field      | Value
|
+-----+
+-----+
+-----+
| created_at   | 2019-03-15T19:42:30Z
| description  |
| direction    | ingress
| ether_type   | IPv4
| id           | c08d31d0-1109-4f78-958e-6ec672c9cb15
| name         | None
| port_range_max | 22
| port_range_min | 22
| project_id   | 95b1989d2c7c42daa1081a8e13d40e16
| protocol     | tcp
| remote_group_id | None
| remote_ip_prefix | 0.0.0.0/0
| revision_number | 0
| security_group_id | c0964b92-d480-4f0f-ab9d-ca69f554469b
| updated_at   | 2019-03-15T19:42:30Z
|
+-----+
+-----+
+-----+
```

- Creamos una clave para poder acceder a la nueva instancia que vamos a crear:

```
stack@ubuntu:~/devstack$ nova keypair-add suriTop8 > suriTop8
```

- Cambiamos los permisos del fichero que contiene la clave:

```
stack@ubuntu:~/devstack$ chmod 400 suriTop_8
```

- Creamos la nueva instancia por comando con el nombre team69-comandos:

```
stack@ubuntu:~/devstack$ openstack server create --flavor dpp --image ubuntuByComandos --  
key-name suriTop8 --security-group defaultByComando --network privateSuri --wait team69-  
comandos
```

Field	Value
OS-DCF:diskConfig	MANUAL
OS-EXT-AZ:availability_zone	nova
OS-EXT-SRV-ATTR:host	ubuntu
OS-EXT-SRV-ATTR:hypervisor_hostname	ubuntu
OS-EXT-SRV-ATTR:instance_name	instance-00000003
OS-EXT-STS:power_state	Running
OS-EXT-STS:task_state	None
OS-EXT-STS:vm_state	active
OS-SRV-USG:launched_at	2019-03-16T10:09:09.000000
OS-SRV-USG:terminated_at	None
accessIPv4	
accessIPv6	
addresses	privateSuri=10.0.0.76
adminPass	MY4sJ4NV5Pbu
config_drive	
created	2019-03-16T10:08:55Z
flavor	suriFlavorByComando (dpp)
hostId	959115e458ad11303276cd9b5f350e2f1eac29d04b9d59f95df288ca
id	b4f8c3b2-094b-450a-a13a-cfd4b9f68422
image	ubuntuByComandos (8bf1ab76-c4f1-43c1-b5c8-b3feb5f64ee0)
key_name	suriTop8
name	team69-comandos
progress	0
project_id	95b1989d2c7c42daa1081a8e13d40e16
properties	
security_groups	name='defaultByComando'
status	ACTIVE
updated	2019-03-16T10:09:10Z
user_id	33e8c9b705a040c3911b1886797d0f41
volumes_attached	

- Comprobamos que la instancia se ha creado correctamente:

```
stack@ubuntu:~/devstack$ openstack server list
```

ID	Name	Status	Networks	Image
Flavor				
b4f8c3b2-094b-450a-a13a-cfd4b9f68422	team69-comandos	ACTIVE	privateSuri=10.0.0.76	
ubuntuByComandos	suriFlavorByComando			
08ab0b45-d8b0-4ed3-8584-fb3e921d4a17	team69	ACTIVE	privateSuri=10.0.0.67,	
172.16.67.231	suriFlavor			

- Creamos una IP pública flotante:

```
stack@ubuntu:~/devstack$ openstack floating ip create public69
```

Field	Value
created_at	2019-03-16T12:30:33Z
description	
fixed_ip_address	None
floating_ip_address	172.16.67.230
floating_network_id	af331bfe-e27e-4de5-a17f-184e81e82e69
id	66efbf1e-b5c2-4cf0-b956-6e5cce36a4eb
name	172.16.67.230
port_id	None
project_id	95b1989d2c7c42daa1081a8e13d40e16
revision_number	0
router_id	None
status	DOWN
updated_at	2019-03-16T12:30:33Z

- Asociamos a la nueva instancia la IP pública creada:

```
stack@ubuntu:~/devstack$ openstack server add floating ip team69-comandos 172.16.67.230
```

- Y por último comprobamos que podemos acceder correctamente a la instancia desde la máquina VMCloud:

```
bigdata@ubuntu:~$ ssh -i suriTop8.pem ubuntu@172.16.67.230
```

Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-108-generic x86_64)

* Documentation: <https://help.ubuntu.com>

* Management: <https://landscape.canonical.com>

* Support: <https://ubuntu.com/advantage>

Get cloud support with Ubuntu Advantage Cloud Guest:

<http://www.ubuntu.com/business/services/cloud>

0 packages can be updated.

0 updates are security updates.

New release '18.04.2 LTS' available.

Run 'do-release-upgrade' to upgrade to it.

Last login: Sat Mar 16 12:33:33 2019 from 172.16.67.225

To run a command as administrator (user "root"), use "sudo <command>".

See "man sudo_root" for details.

Al igual que en el ejercicio 7 procedemos mediante la herramienta sysbench y el crontab a monitorizar el uso de CPU, para esta instancia. Adjuntamos el fichero cpu-sysbench a los documentos de la práctica.

```
ubuntu@team69-comandos:~$ nproc
1
```

```
ubuntu@team69-comandos:~$ free -m
      total        used        free      shared  buff/cache   available
Mem:      968         51        713           3        203        768
Swap:      0           0           0
```

- Creamos un script para que se ejecute la herramienta sysbench y que se compruebe el test de la CPU solicitado:

```
ubuntu@team69-comandos:~/$ pico /home/ubuntu/scripts/cpu-sysbench8.sh
# !/bin/bash
sysbench --test=cpu --cpu-max-prime=20000 --num-threads=2 run
```

- A continuación actualizamos nuestro gestor de paquetes e instalamos la herramienta:

```
ubuntu@team69:~/$ sudo apt-get update
ubuntu@team69:~/$ sudo apt-get install sysbench
```

- Después introducimos una tarea programada en el cron del sistema para que se ejecute cada 2 min y lance el script creado y guarde la salida en el fichero cpu-sysbench8.out:

```
ubuntu@team69-comandos:~$ crontab -e
Select an editor. To change later, run 'select-editor'.
 1. /bin/ed
 2. /bin/nano    <---- easiest
 3. /usr/bin/vim.basic
 4. /usr/bin/vim.tiny
```

```
Choose 1-4 [2]: 2
crontab: installing new crontab
```

```
# Edit this file to introduce tasks to be run by cron.
# ...
```

```
*/2 * * * * root bash /home/ubuntu/scripts/cpu-sysbench8.sh >> /home/ubuntu/cpu-sysbench8.out
```

```
ubuntu@team69:~$ crontab -e
crontab: installing new crontab
```


Después de realizar la monitorización, redimensionamos la instancia y le asignamos 2 CPUs, 1GB de RAM y el mismo disco.

Para llevar a cabo esto tenemos que tener en cuenta lo siguiente: hay que tener cuidado con redimensionar un flavor cuando está siendo usado en alguna instancia. Esto es una issue de algunas versiones y al hacerlo hace que la instancia pierda la información de la imagen y el flavor asociado.

Por eso lo mejor es crear un nuevo flavor y a la hora de redimensionar la instancia cambiar el antiguo por el nuevo.

- Creamos un nuevo flavor:

```
stack@ubuntu:~/devstack$ nova flavor-create suri8+ dpp8 1000 3 2
```

ID	Name	Memory_MB	Disk	Ephemeral	Swap	VCPUs	RXTX_Factor	Is_Public
dpp8	suri8+	1000	3	0		2	1.0	True

- A continuación redimensionamos la instancia asociándole el nuevo flavor creado:

```
stack@ubuntu:~/devstack$ openstack server resize --flavor suri8+ --wait suri-santi-comandos  
Complete
```

- Por último confirmamos la redimensión:

```
stack@ubuntu:~/devstack$ openstack server resize --confirm suri-santi-comandos
```

Para comprobar que la redimensión se ha llevado a cabo accedemos a la máquina y comprobamos el número de CPUs y RAM que tiene la instancia:

```
bigdata@ubuntu:~$ ssh -i suriTop8.pem ubuntu@172.16.67.230
```

```
ubuntu@team69-comandos:~$ free -m
```

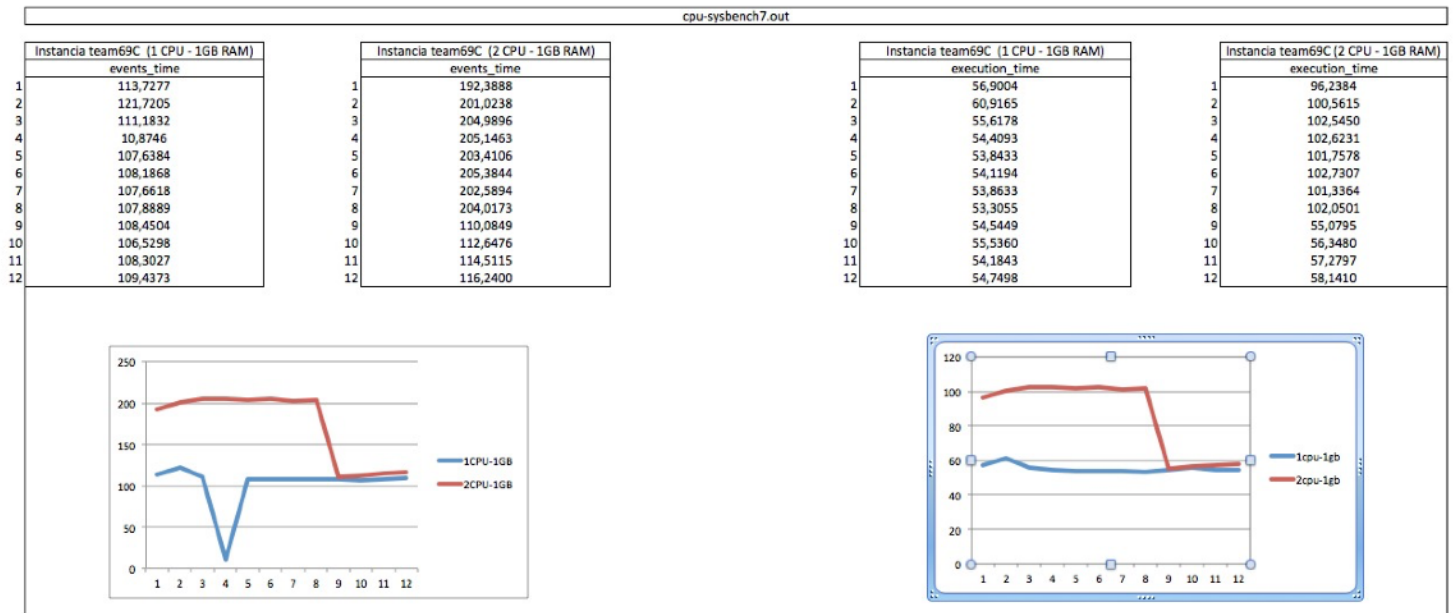
	total	used	free	shared	buff/cache	available
Mem:	968	75	571	2	320	741
Swap:	0	0	0			

```
ubuntu@team69-comandos:~$ nproc
```

```
2
```

Podemos ver como se ha incrementado el número de CPUs, como habíamos puesto en el nuevo flavor.

A continuación pasamos a recoger la traza después de este cambio en el fichero cpu-sysbench-8.out, para poder hacer las comparativas y los conclusiones solicitadas.



Al igual que en el ejercicio anterior, una vez ejecutado esto se genera el fichero cpu-sysbech8.out que contiene todas las trazas de la primera fase y la segunda, una vez redimensionada la instancia, lo que nos permite hacer los análisis y conclusiones solicitados en el ejercicio:

Los tiempos obtenidos en la primera fase con una instancia de 1 CPU y 1GB de RAM son menores que cuando redimensionamos la instancia y le asignamos 2 CPUs. Esto no tiene en principio mucho sentido, pues si la máquina tiene más cores, la lógica nos haría pensar que tiene más capacidad de procesamiento y que en el segundo caso la CPU debería de tardar menos tiempo en realizar los test, pero esto no ocurre así.

Hemos investigado más sobre este caso y puede ser que la instalación que hemos hecho no tenga la configuración correcta o no soporte KVM anidados como comentas en este hilo oficial de OpenStack: <https://ask.openstack.org/en/question/94405/very-slow-cpu-for-compute-instances-on-tripleo-in-virtual-environment/>