Grupo 2402, pareja 2
Lucía Colmenarejo Pérez
Luis Cárabe

SI 2
Práctica 2: Rendimiento

Ejercicio 1: siguiendo todos los pasos anteriores, defina el plan completo de pruebas para realizar las tres ejecuciones secuenciales sobre los tres proyectos definidos hasta ahora (P1-base, P1-ws, P1-ejb). Adjunte el fichero generado P2.jmx al entregable de la práctica.

Hemos completado todo el plan de pruebas siguiendo el guión, simplemente cabe destacar la creación de tres variables idTransaccion distintos, para que no haya ningún posible colapso, llamadas *idTransaccion1*, *idTransaccion2* e *idTransaccion3*, para *P1-base*, *P1-ws* y *P1-ejb* respectivamente.

Ejercicio 2: preparar los PCs con el esquema descrito en la Figura 21.

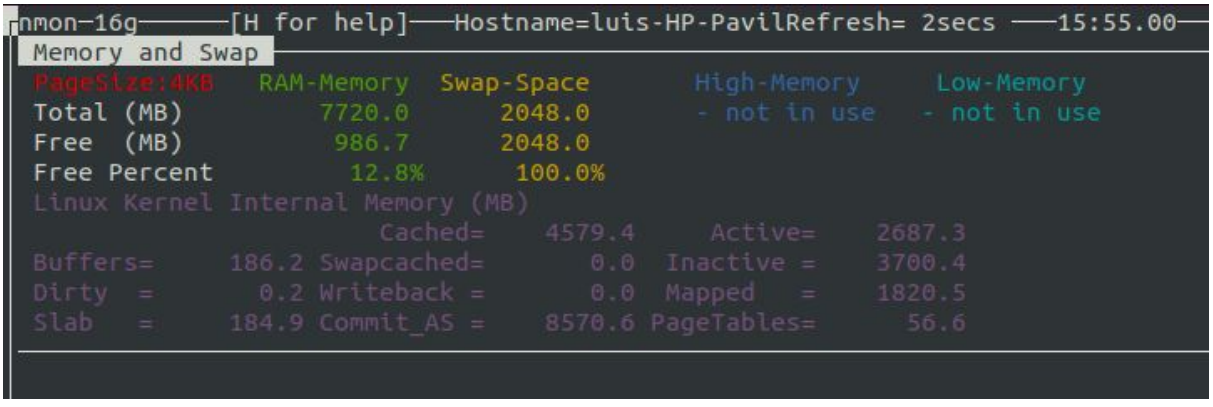
Para este ejercicio hemos ido realizando todos los pasos indicados, teniendo en cuenta que para facilitarnos el trabajo fuera de los horarios lectivos, hemos decidido realizar la práctica en un único ordenador. Por ello, un único PC será el encargado de dar soporte a las dos máquinas virtuales.

Tras modificar los ficheros de propiedades de todas las prácticas y desplegarlas correctamente, ejecutamos el comando *free* en el PC:

```
luis@luis-HP-Pavilion-x360:~/Desktop/UAM/SI2/SI2/P2-alumnos$ free
              total        used        free      shared  buff/cache   available
Mem:        7905324     1888760     1011256     2029188     5005308     3701356
Swap:       2097148           0     2097148
```

Podemos observar como de los 8gb de memoria RAM que dispone nuestro portátil, apenas tenemos un gb sin usar, no obstante, 3.7 gb están a nuestra disposición. El resto, está siendo ocupada por las máquinas virtuales y por las aplicaciones del propio PC.

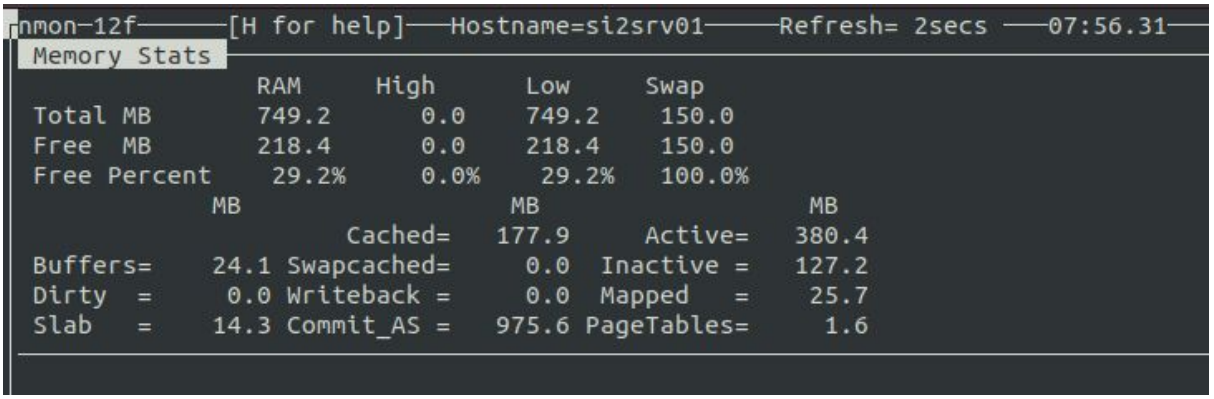
A continuación mostramos la salida del comando *nmon*:

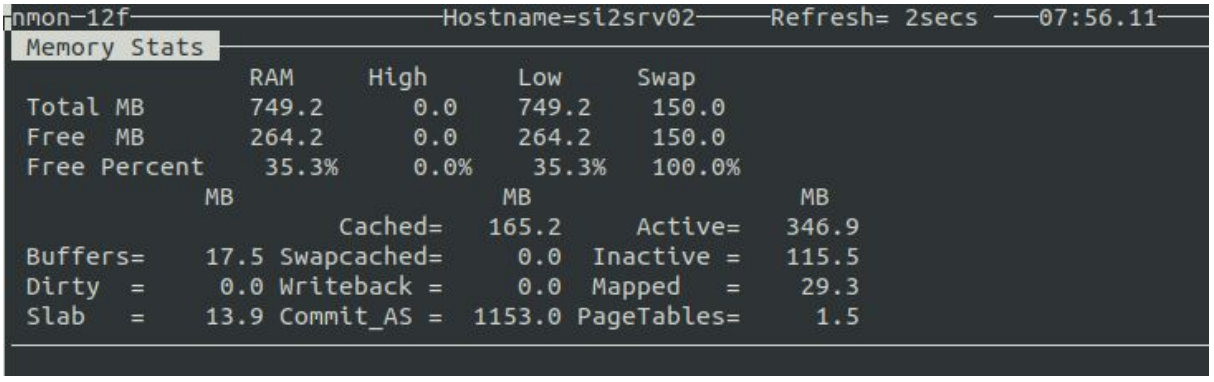


La información que nos ofrece es bastante similar, en este caso, *nmon* se refiere a la memoria *available* como *inactive*, pero vemos que los valores son los mismos.

Por tanto, tenemos la capacidad más que suficiente para realizar todas las pruebas necesarias sin que los recursos de memoria escaseen.

Las siguientes capturas se corresponden con el comando *nmon* invocado desde las dos máquinas virtuales (primero *si2srv01* y después *si2srv02*):



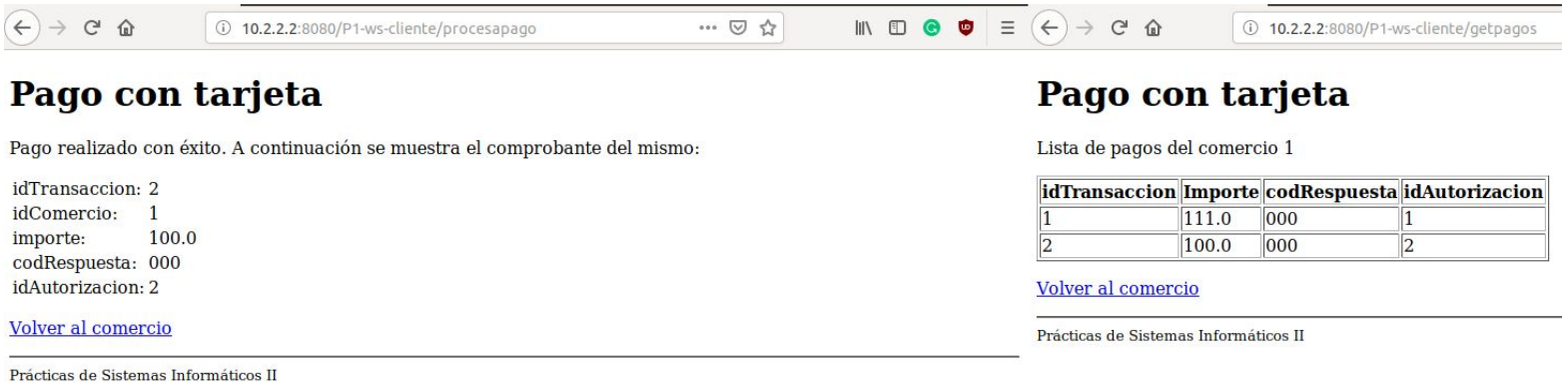


Ambas arrojan unos resultados similares, teniendo 750 mb de ram en total, con 115/127 mb inactivos y unos 218/265 mb sin usar. Solamente notamos una mayor diferencia en la RAM activa, siendo mayor en la máquina 1.

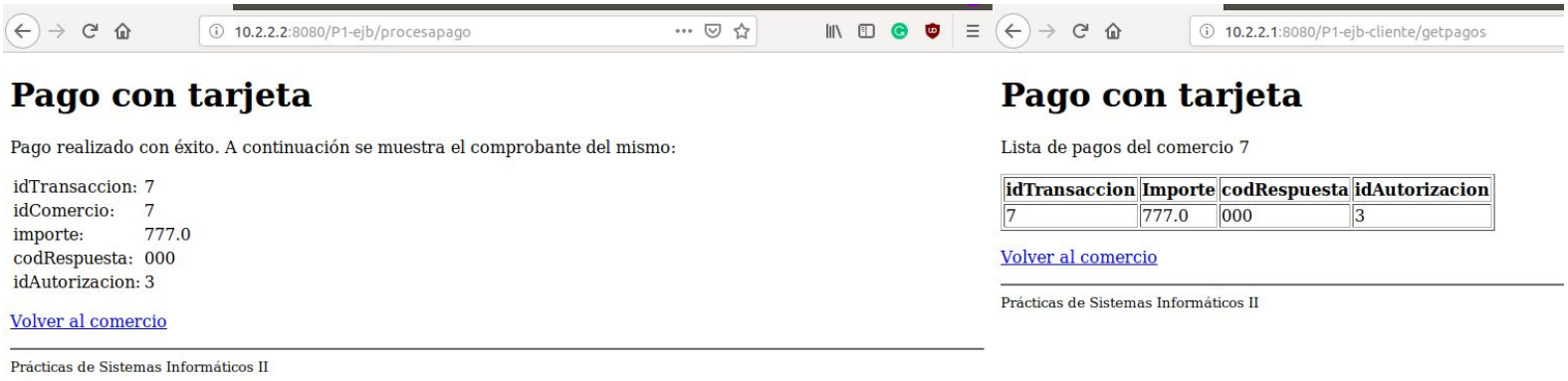
Ahora presentaremos los pagos de prueba realizados para comprobar el correcto funcionamiento de las aplicaciones desplegadas, a la izquierda el comprobante del pago, a la derecha la comprobación mediante *get pagos* de *testbd.jsp*:

P1-base:

P1-ws:



P1-ejb:



Ejercicio 3: ejecute el plan completo de pruebas sobre las 3 versiones de la práctica, empleando el esquema de despliegue descrito anteriormente.

Tras realizar la prueba varias veces, cogemos el resultado de una que es la que menos difiere con el resto, es decir, la más representativa que elimina de mejor manera el ruido:

Label	# Samples	Average	Median	90% Line	95% Line	99% Line
P1-base	1000	4	5	5	6	9
P1-ws	1000	39	38	44	51	68
P1-ejb	1000	11	11	13	14	21
TOTAL	3000	18	11	39	42	61

Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
4	26	0.00%	198.3/sec	257.15	85.09
34	80	0.00%	25.1/sec	32.82	10.91
9	46	0.00%	85.3/sec	110.40	36.51
4	80	0.00%	53.0/sec	68.86	22.82

Vemos como tenemos un 0% de error. Además, realizamos la consulta SQL propuesta y, efectivamente, obtenemos 3000 pagos.

Adjuntamos, en la carpeta *Adjuntos ejercicio 3*, los dos server logs, el de la máquina virtual 10.2.2.1 como server1 y el de la máquina 10.2.2.2 como server2.

¿Cuál de los resultados le parece el mejor? ¿Por qué? ¿Qué columna o columnas elegiría para decidir este resultado?

Desde nuestro punto de vista, todos los parámetros analizados parecen estar muy relacionados, no obstante, creemos que el más característico es el *throughput*, es decir, el número de respuestas a eventos que se realizan por unidad de tiempo que se observan en un intervalo, ya que es un parámetro muy representativo para medir el rendimiento de una aplicación. Con esto, consideramos que el mejor resultado lo consigue *P1-base*, con un *throughput* de 193.1/seg, muy lejano al de *P1-ejb* de 85.3/seg y aún más de *P1-ws* (25.1/s). Y aún considerando cualquier otra columna de la tabla, *P1-base* es el que mejor rendimiento presenta.

Otra columna representativa sería la latencia al 90% (*90% line*), cuanto menor es el número, menor es la espera para obtener respuesta, vemos que *P1-base* resulta ganador también.

Ahora inhabilitamos P1-base y P1-ws para poder probar P1-ejb con EJB local, los resultados que arroja JMeter son los siguientes:

Label	# Samples	Average	Median	90% Line	95% Line	99% Line
P1-ejb	1000	3	4	4	5	7
TOTAL	1000	3	4	4	5	7

Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
3	35	0.00%	250.1/sec	327.96	109.03
3	35	0.00%	250.1/sec	327.96	109.03

Vemos que la diferencia de rendimiento es muy notoria, poniéndose casi a la par que *P1-base* en términos de *throughput*, por tanto, el EJB local provoca una mejora significante en el rendimiento, multiplicándolo por diez.

Ejercicio 4: adaptar y guardar la configuración del servidor de aplicaciones. Ejecutar los mandatos asadmin necesarios para obtener los valores de Max Queue Size del Servicio HTTP y Maximum Pool Size del Pool de conexiones a nuestra DB. Así como el mandato para monitorizar el número de errores en las peticiones al servidor web.

Junto con esta memoria se adjunta el fichero *domain.xml* que contiene la configuración resultante de realizar las modificaciones explicadas en el enunciado, el cual hemos obtenido parando el servidor de aplicaciones y accediendo a la ruta `$opt/glassfish4/glassfish/domains/domain1/config/domain.xml` en la máquina virtual.

Los mandatos asadmin que hemos realizado para obtener los datos requeridos en el enunciado son los siguientes:

```
luis@luis-HP-Pavilion-x360:~/Desktop/UAM/SI2/SI2/P2-alumnos$ asadmin --host 10.2.2.2 --user admin --passwordfile ./passwordfile get server.thread-pools.thread
-pool.http-thread-pool.max-queue-size
server.thread-pools.thread-pool.http-thread-pool.max-queue-size=4096
Command get executed successfully.
luis@luis-HP-Pavilion-x360:~/Desktop/UAM/SI2/SI2/P2-alumnos$ asadmin --host 10.2.2.2 --user admin --passwordfile ./passwordfile get resources.jdbc-connection-
pool.VisaPool.max-pool-size
resources.jdbc-connection-pool.VisaPool.max-pool-size=32
Command get executed successfully.
luis@luis-HP-Pavilion-x360:~/Desktop/UAM/SI2/SI2/P2-alumnos$ asadmin --host 10.2.2.2 --user admin --passwordfile ./passwordfile monitor --type httplistener
ec   mt   pt     rc
0    0    0.00   0
0    0    0.00   0
```

En el último de los comandos ejecutados observamos el número de errores al realizar las peticiones al servidor web se observa en la primera columna “ec” (*error count*).

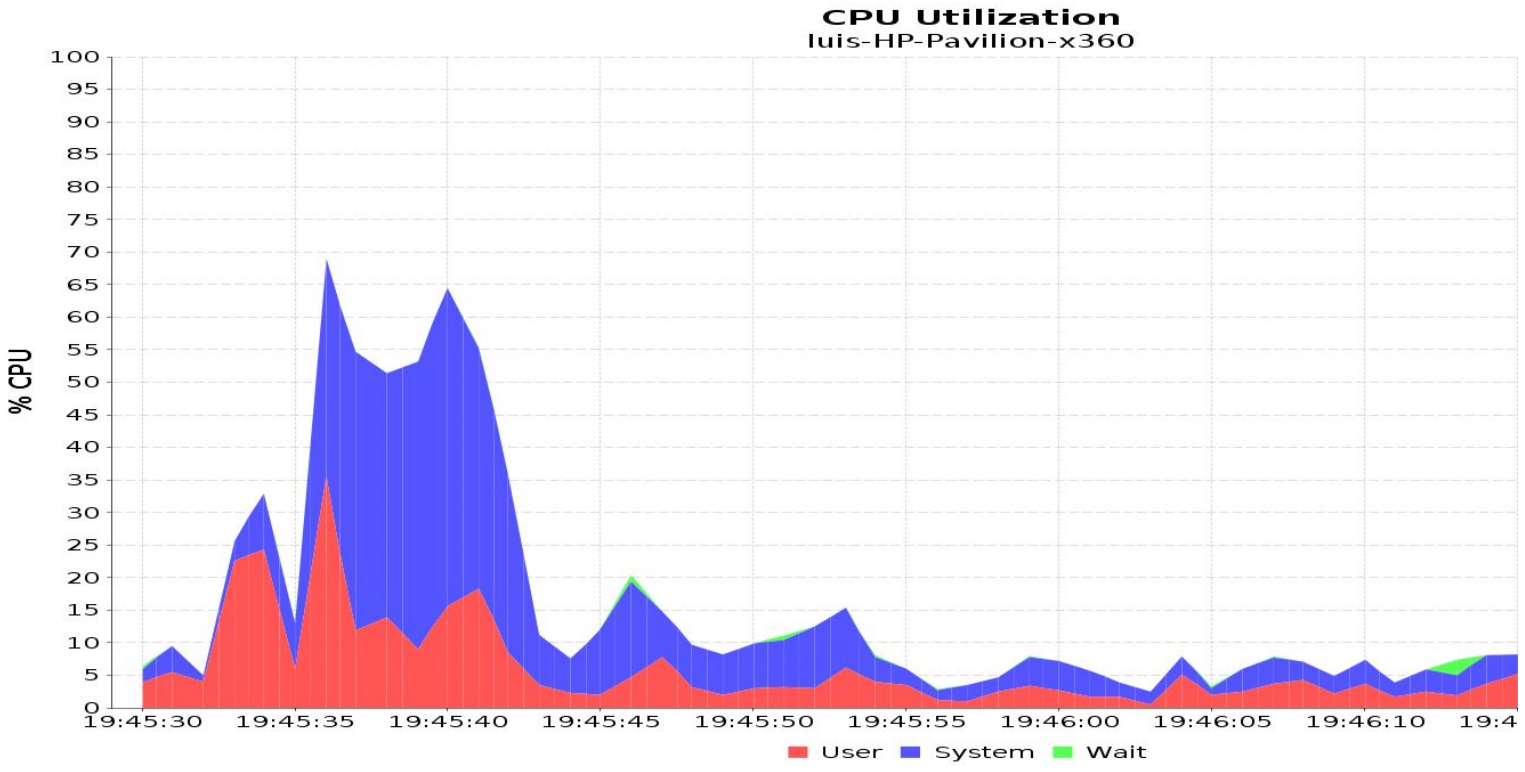
Ejercicio 5: registrar en la hoja de cálculo de resultados los valores de configuración que tienen estos parámetros.

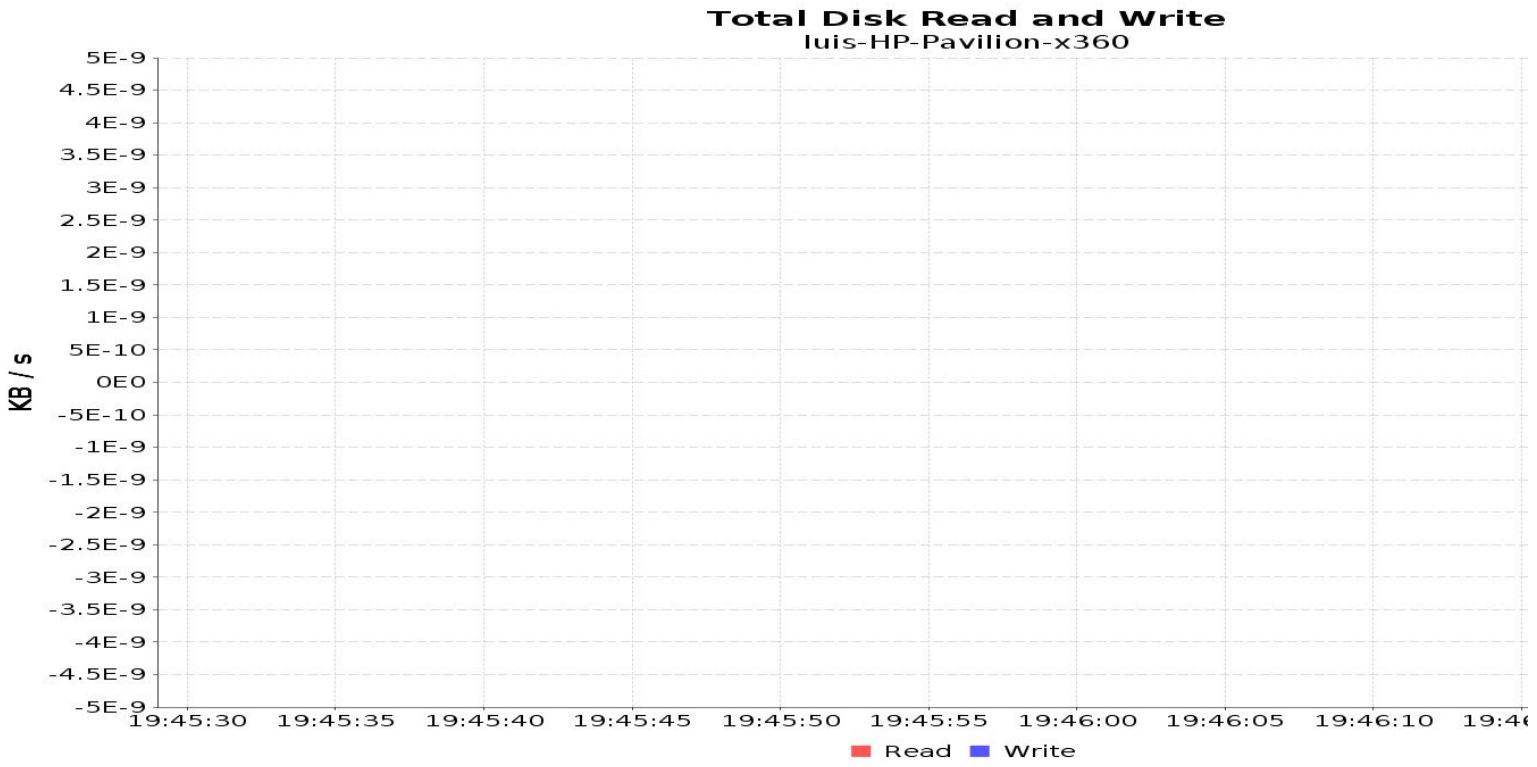
Se podrá ver en el fichero *SI2-P2-curvaProductividad* entregado en la práctica.

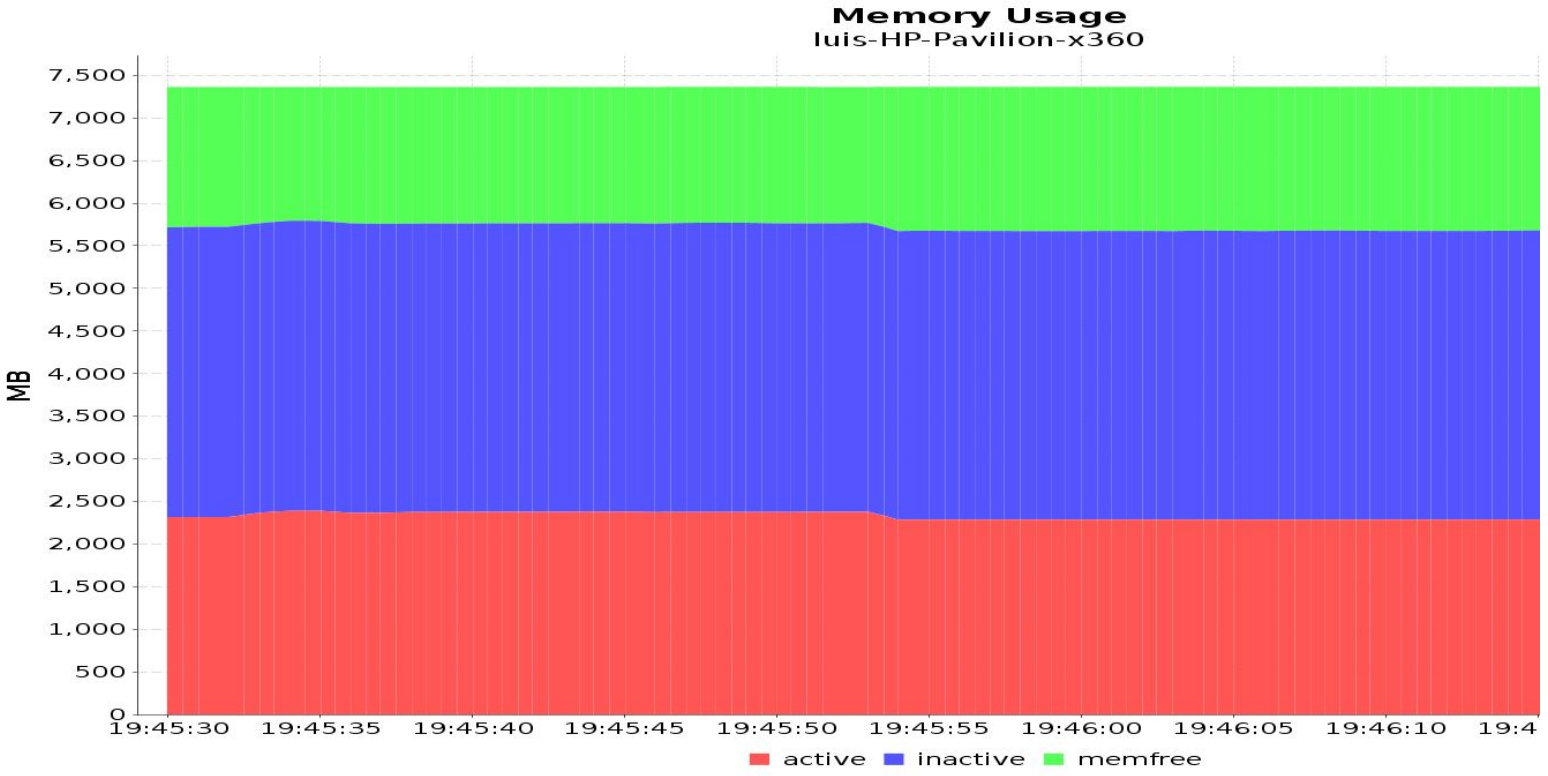
Ejercicio 6: tras habilitar la monitorización en el servidor, repita la ejecución del plan de pruebas anterior. Durante la prueba, vigile cada uno de los elementos de monitorización descritos hasta ahora.Responda a las siguientes cuestiones:

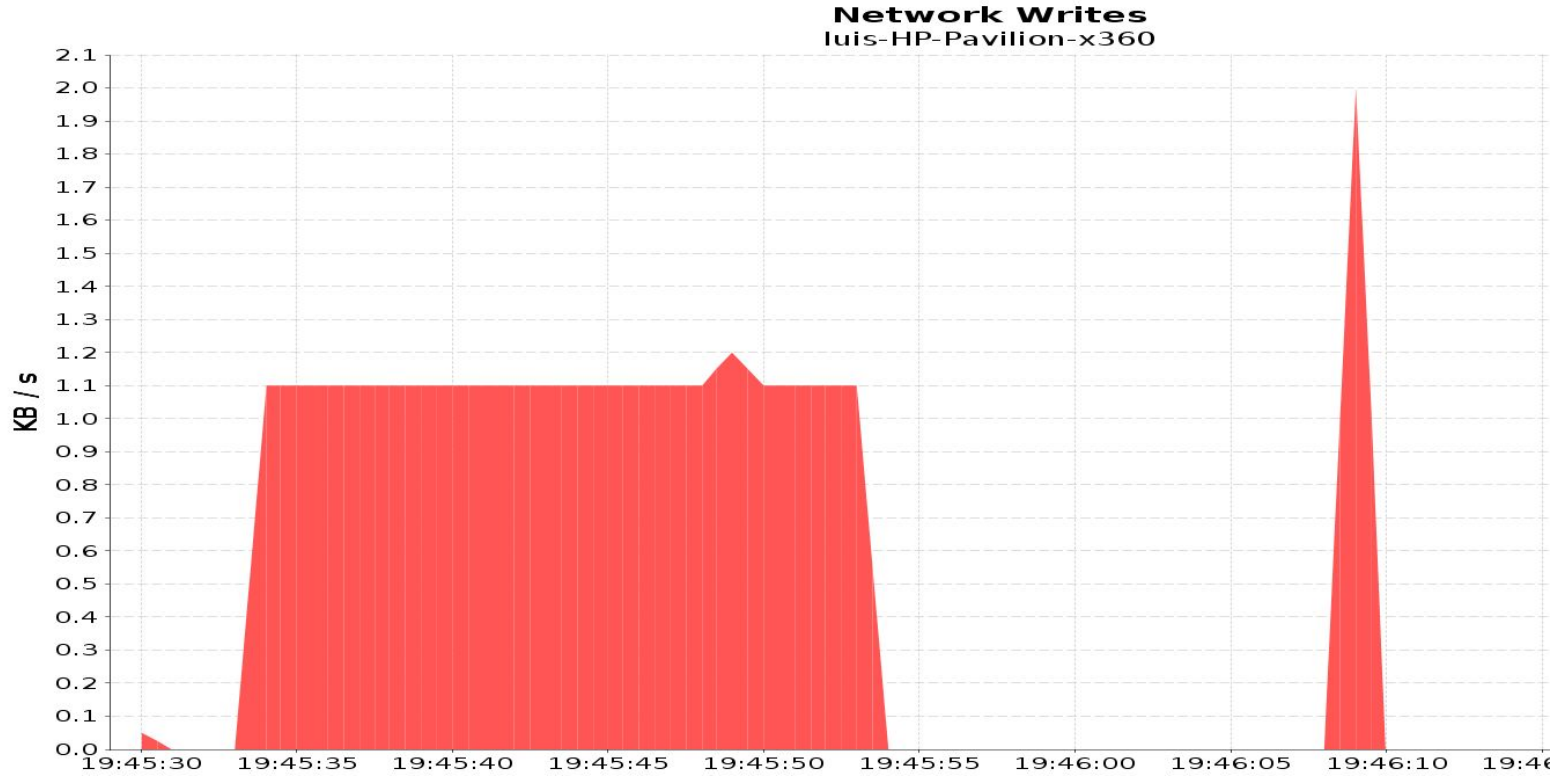
Antes de contestar a las cuestiones, vamos a presentar las gráficas correspondientes.

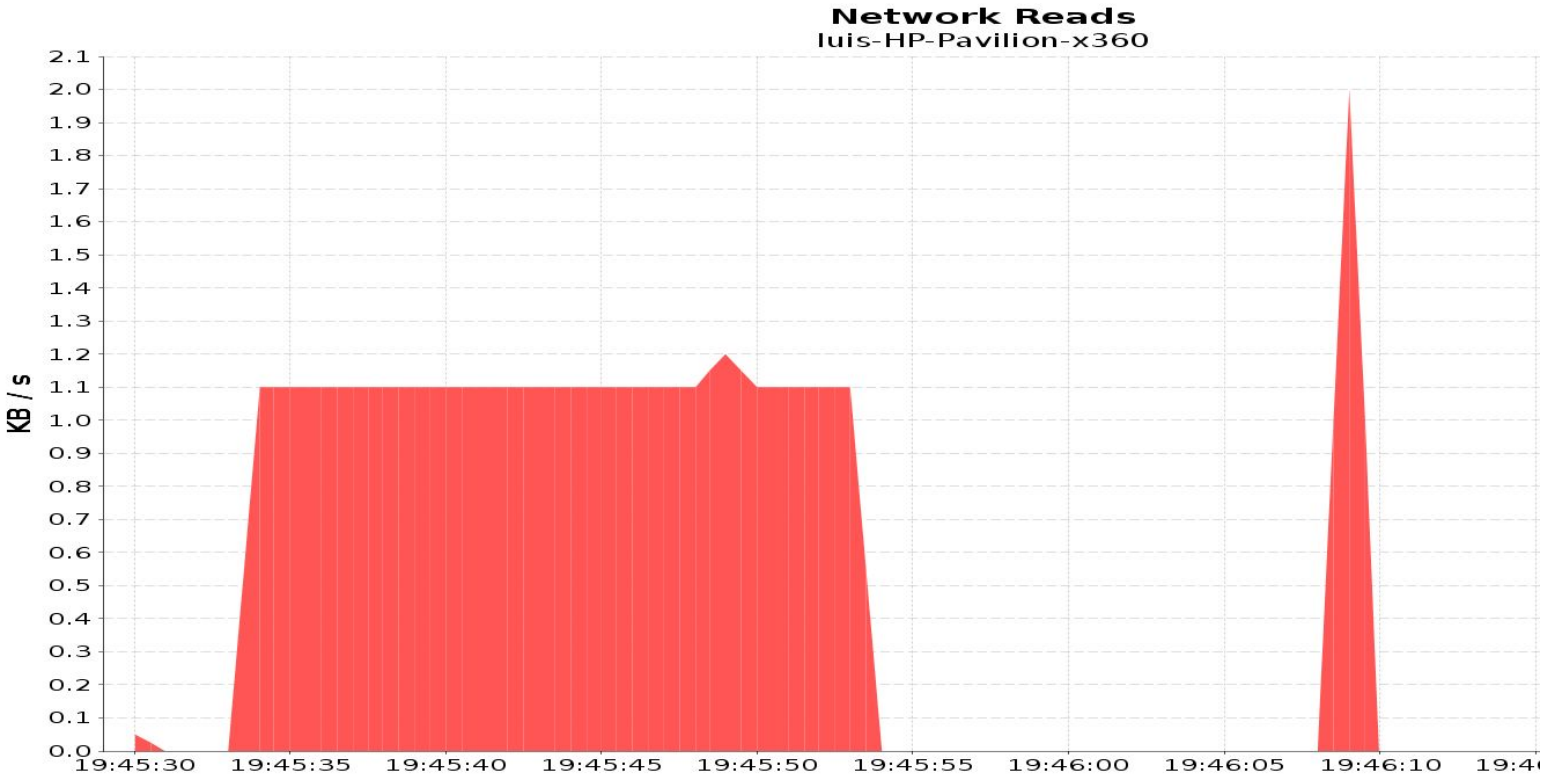
Uso de los recursos del PC:



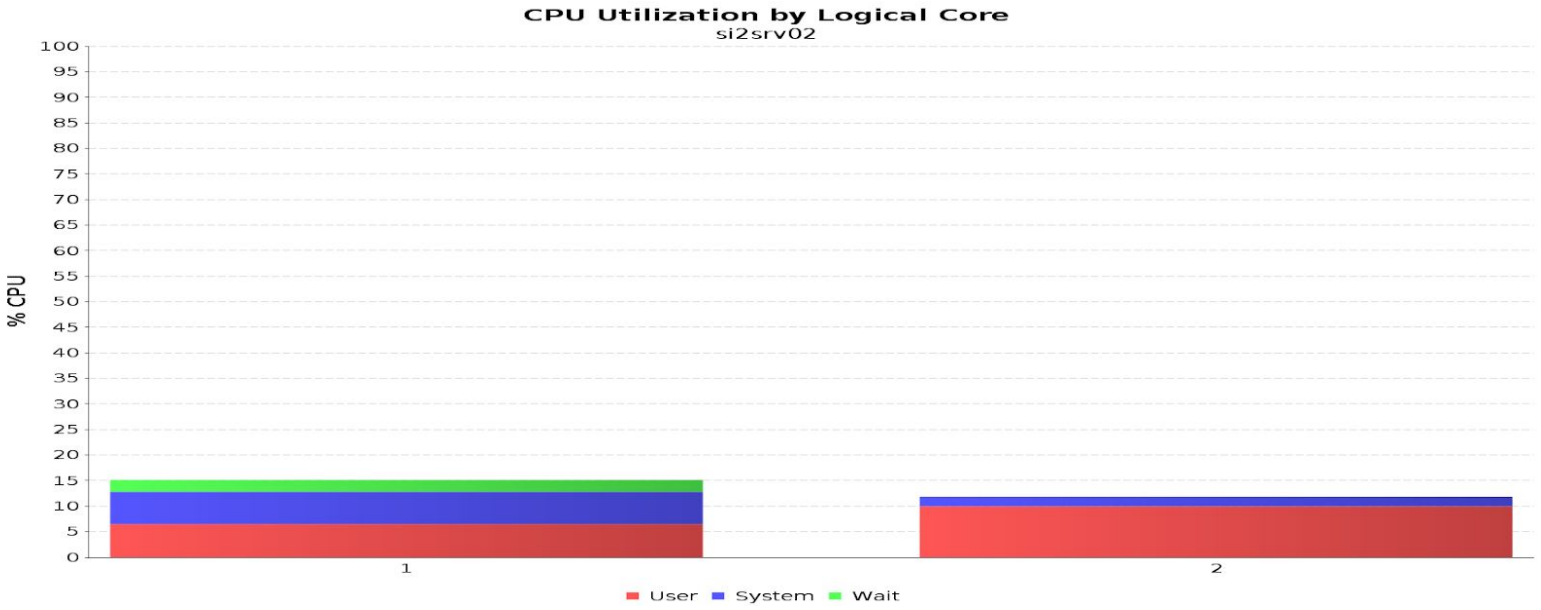

Total Disk Read and Write
luis-HP-Pavilion-x360
KB / s
5E-9
4.5E-9
4E-9
3.5E-9
3E-9
2.5E-9
2E-9
1.5E-9
1E-9
5E-10
0E0
-5E-10
-1E-9
-1.5E-9
-2E-9
-2.5E-9
-3E-9
-3.5E-9
-4E-9
-4.5E-9
-5E-9
19:45:30
19:45:35
19:45:40
19:45:45
19:45:50
19:45:55
19:46:00
19:46:05
19:46:10
Read
Write

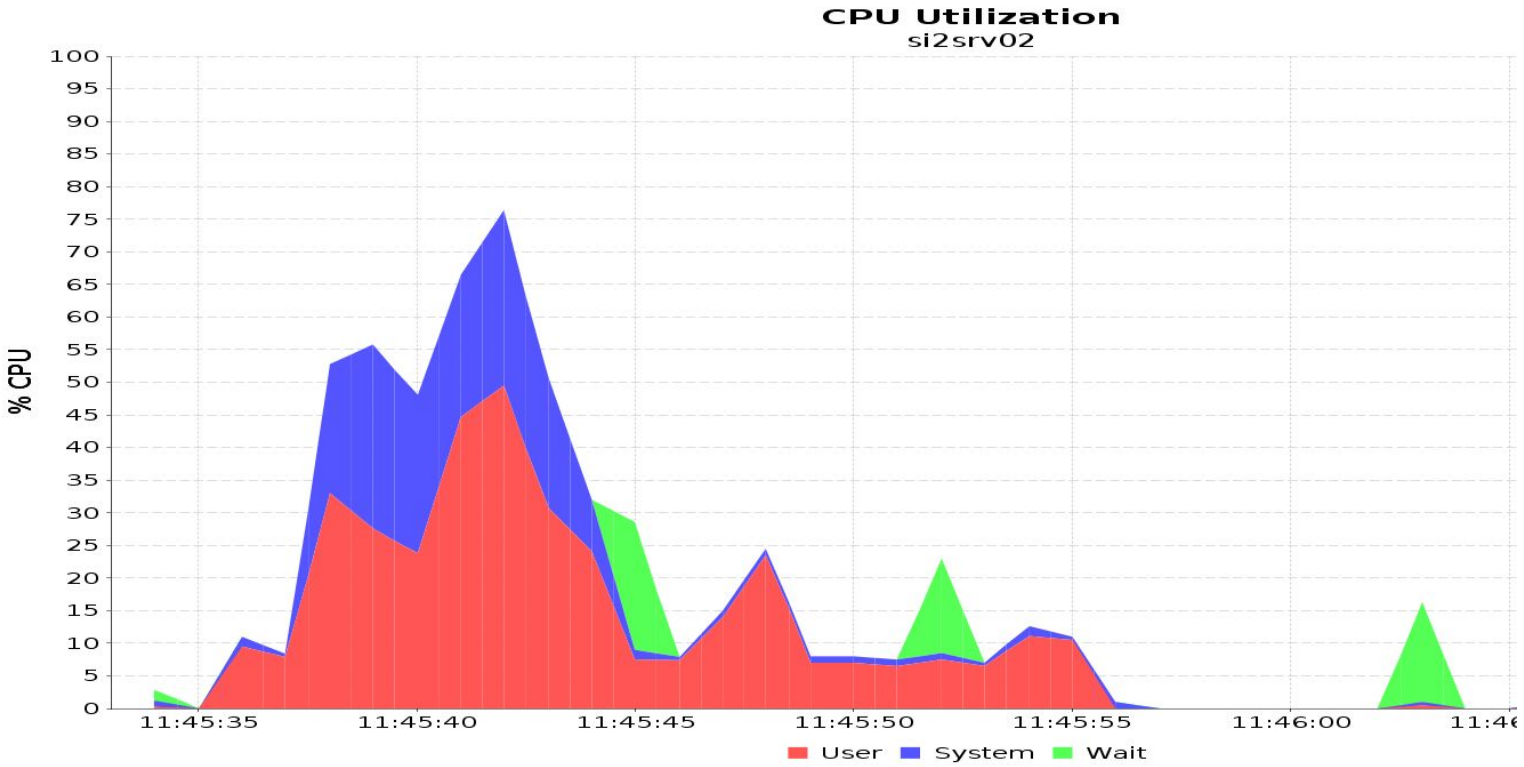

Memory Usage
luis-HP-Pavilion-x360
MB
7,500
7,000
6,500
6,000
5,500
5,000
4,500
4,000
3,500
3,000
2,500
2,000
1,500
1,000
500
0
19:45:30
19:45:35
19:45:40
19:45:45
19:45:50
19:45:55
19:46:00
19:46:05
19:46:10
19:4
active
inactive
memfree

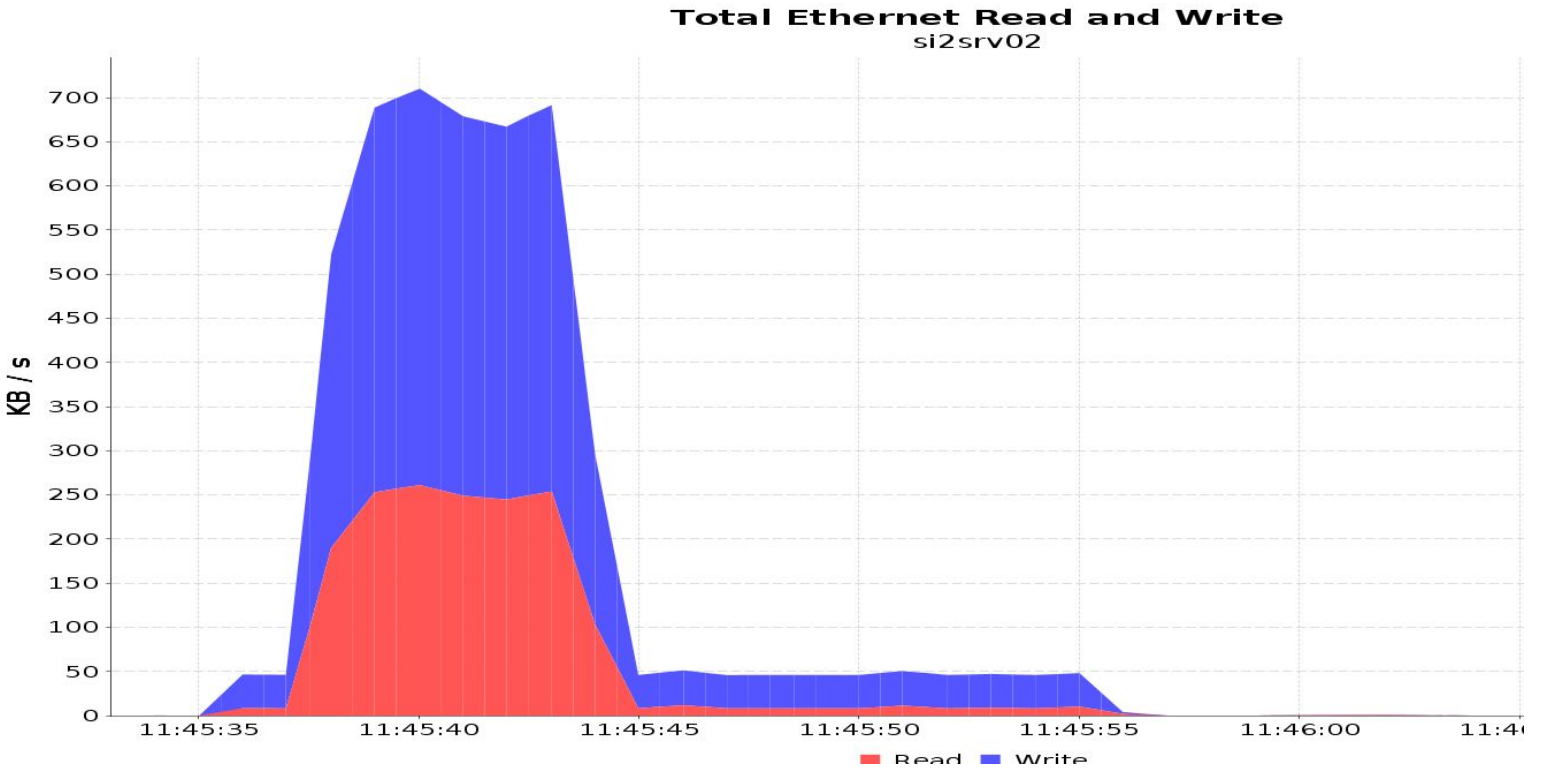

Network Writes
luis-HP-Pavilion-x360
KB / s
2.1
2.0
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
19:45:30
19:45:35
19:45:40
19:45:45
19:45:50
19:45:55
19:46:00
19:46:05
19:46:10
19:46

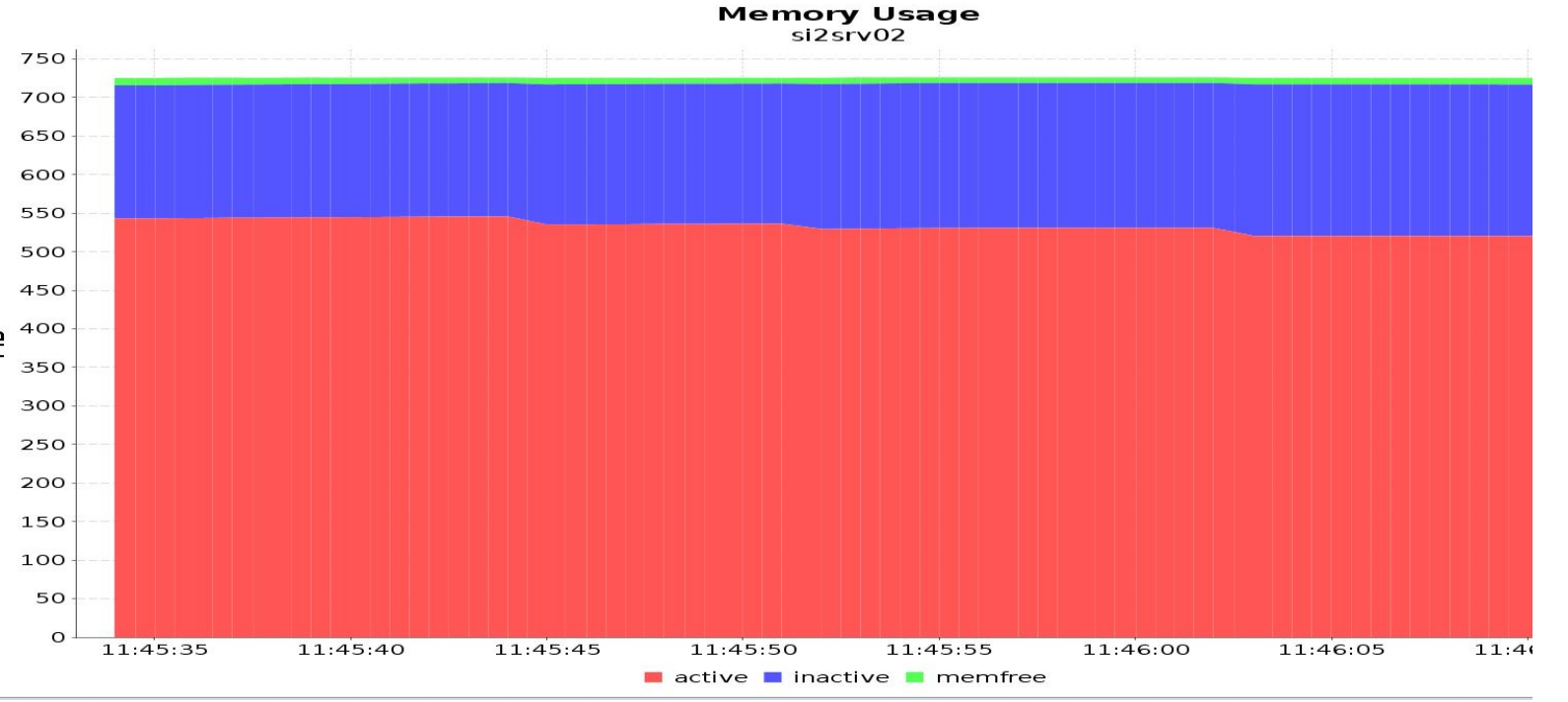


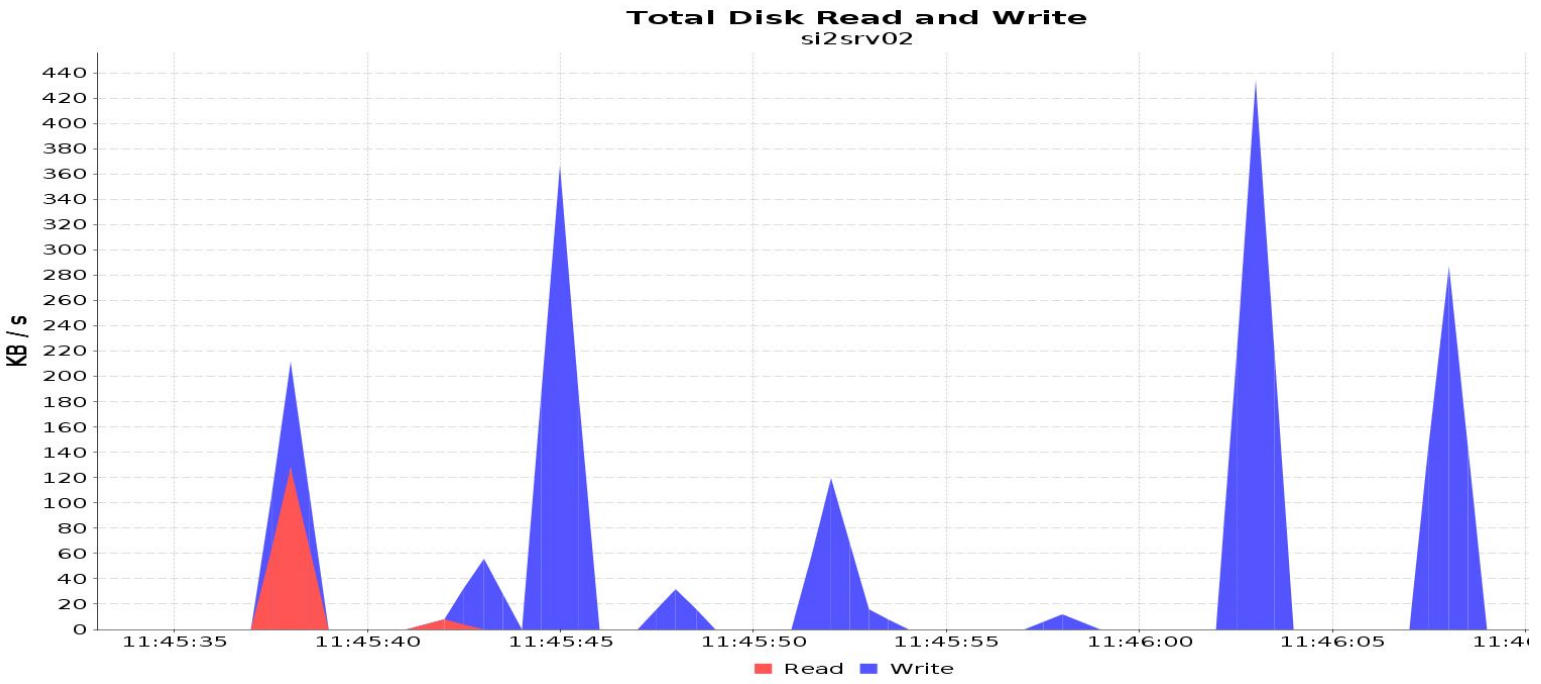
Uso de los recursos de la máquina virtual (cabe destacar que cuenta con dos núcleos de CPU):


CPU Utilization by Logical Core
si2srv02
% CPU
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
1
2
User
System
Wait


CPU Utilization
si2srv02
% CPU
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
11:45:35
11:45:40
11:45:45
11:45:50
11:45:55
11:46:00
User
System
Wait


Total Ethernet Read and Write
si2srv02
KB / s
700
650
600
550
500
450
400
350
300
250
200
150
100
50
0
11:45:35
11:45:40
11:45:45
11:45:50
11:45:55
11:46:00
Read
Write


Memory Usage
si2srv02
750
700
650
600
550
500
450
400
350
300
250
200
150
100
50
0
11:45:35
11:45:40
11:45:45
11:45:50
11:45:55
11:46:00
11:46:05
active
inactive
memfree



Por último, adjuntamos la salida del script *si2-monitor-sh*:

#Muestra	numJDBCCount	numHTTPCount	numHTTPQ
0	0	0	0
1	0	0	0
2	1	1	0
3	1	1	0
4	0	1	0
5	0	1	0
6	0	1	0
7	1	1	0
8	1	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0
16	0	0	0
17	0	0	0
18	0	0	0
19	0	0	0
TOT.MUESTRAS	MEDIA:		
20	0.2	0.3	0

El resultado de este script nos asegura que el servidor está muy desahogado, ya que nunca llega a haber elementos en la cola (numHTTPQ) y sus recursos, como son el pool de conexiones y el número de hilos en uso está muy próximo a 0.

- **A la vista de los resultados, ¿qué elemento de proceso le parece más costoso? ¿Red? ¿CPU? ¿Acceso a datos? En otras palabras, ¿cuál fue el elemento más utilizado durante la monitorización con nmon en un entorno virtual? (CPU, Memoria, disco...)**

  Vemos que realmente, ningún recurso llega a saturarse, no obstante, en el PC, destaca un uso de la CPU (con un pico del 70%) y una velocidad de lectura/escritura en red de 1.1 kB/s. En la máquina virtual pasa algo parecido, alcanzando un uso de CPU del 75% y un uso de red, ahora mucho mayor en escritura que en lectura, llegando a los 700 kB/s, esto es debido a que realiza el doble de consultas de las que recibe. Cabe destacar también un uso de escritura en disco, con un pico de 440 kB/s. La memoria no parece verse afectada de manera notoria en ninguno de los dos casos.

  Por tanto, el recurso más costoso sería la Red y la CPU.

- **¿Le parece una situación realista la simulada en este ejercicio? ¿Por qué?**

  No, ya que no contamos con el *think time*, tiempo que un usuario normal pasaría sin interactuar con la página, en cambio, nuestra simulación ejecuta muchas peticiones sin ninguna pausa, perdiendo fiabilidad para transportar los resultados a la vida real.

- **Teniendo en cuenta cuál ha sido el elemento más saturado, proponga otro esquema de despliegue que resuelva esa situación**

  Teniendo en cuenta que las consultas a la base de datos es la causante de la alta utilización de la red, un remedio sencillo sería traspasar la base de datos a la misma dirección desde donde hacemos las consultas, es decir la dirección 10.2.2.2, no obstante, este modelo resulta algo impensable en la vida real, ya que las personas que actúen como cliente, si llegan a tener una copia de la base de datos cada una, sería complicado y costoso mantenerlas todas coherentes entre sí (con los mismo datos). Para el elevado uso del CPU no creemos que tenga fácil arreglo simplemente cambiando el esquema de despliegue.

Ejercicio 8: obtener la curva de productividad

Realizando la prueba hemos tomado valores de clientes que varían de 250 a 11000, que es cuando dejamos de apreciar incremento del *throughput*. En la práctica, hemos probado con más valores de los representados en la tabla adjunta, no obstante, no aportaban ningún tipo de información y al ir realizando pruebas nos dimos cuenta que debíamos dejar más espacio de número de usuarios entre unas pruebas y otras. Por tanto, aunque no están representados, en la carpeta *Adjuntos ejercicio 8*, podemos encontrar los

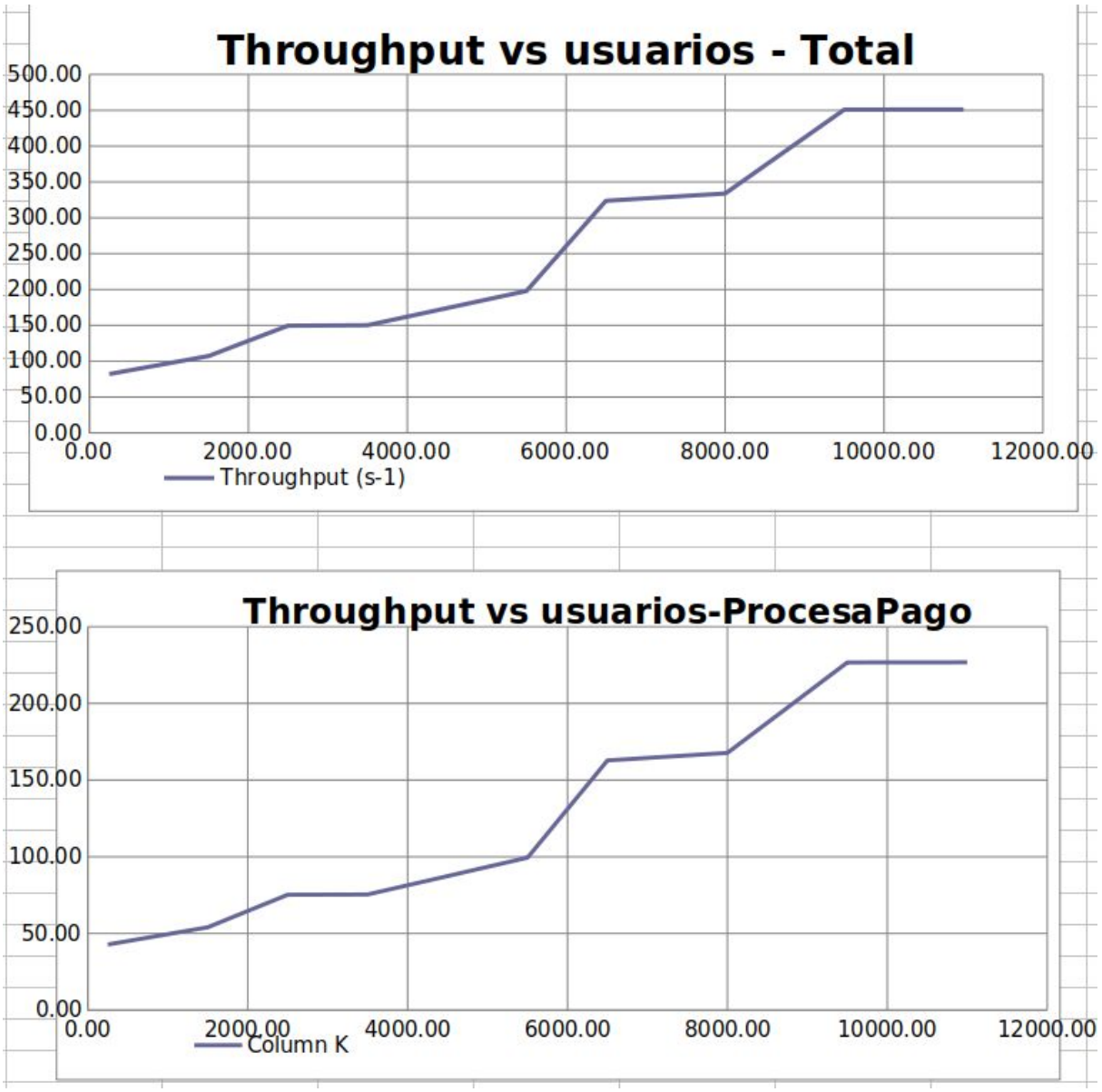
ficheros de datos de las pruebas de valores como 500, 750, 3000, 4000… que no están representados en la tabla por dejar más espacio a números más altos. Para entender bien estos ficheros es necesario explicar cómo están guardados:

- *XXXX.csv*: archivos que guardan las tablas resultantes del JMeter.
- *monitorXXXX.dat*: archivos que guardan los resultados del script *si-monitor.sh*.
- *vmstatXXXX.dat*: archivos que guardan los resultados del comando vmstat.

En todos ellos, XXXX representa el número de usuarios. La monitorización del PC físico la hemos realizado dinámicamente, con nmon abierto, cerciorándonos en todo momento que la CPU no se saturara, no lo hizo en ningún momento, podemos verlo también en los porcentajes de uso de la CPU por la máquina virtual, como apenas llegan al 65%.

Al llegar a 11000 usuarios observamos como el script *si-monitor.sh* deja de funcionar al estar demasiado saturada la máquina. No obstante, el throughput no cambia con respecto a 9500 usuarios, por tanto, preferimos no seguir subiendo número de usuarios, creemos con mucha fuerza que el *throughput* no va a ascender.

Adjuntamos ahora capturas de las tablas de throughput para mayor comodidad en el ejercicio 9.



Ejercicio 9: responda a las siguientes cuestiones:

- **A partir de la curva obtenida, determinar para cuántos usuarios conectados se produce el punto de saturación, cuál es el throughput que se alcanza en ese punto, y cuál el throughput máximo que se obtiene en zona de saturación.**

  Hemos intentado hacer la práctica en un entorno estable, en casa, a una hora en la que no hubiera ninguna persona conectada a dicha red, no obstante, con los datos que hemos obtenido es difícil tener claras las distintas regiones. Parecería que la lineal va de 250 a 6500, pero si trazamos una línea que pasen por dichos puntos, casi toca en el punto 9500, por tanto, si nos basamos en los resultados empíricos, el punto de saturación se alcanza en el propio 9500 (usuarios), con un *throughput* de 450/s, siendo este a su vez el throughput máximo.

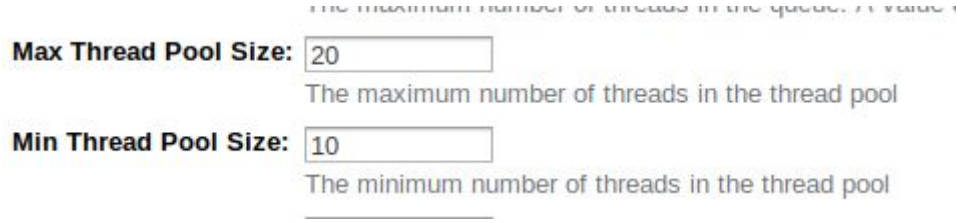
No obstante, no podemos ignorar a nuestra intuición, que nos hace ver que a la fuerza debe existir una zona de transición, difusa en las pruebas. Esta zona, de existir, parecería estar entre 2500 y 9500. La zona lineal, en este caso, resultaría de unir el punto (0,0) con el punto para 2500 usuarios. Esto nos daría un punto de saturación en 6500 usuarios, con un throughput de 320/s.

- **Analizando los valores de monitorización que se han ido obteniendo durante la elaboración de la curva, sugerir el parámetro del servidor de aplicaciones que se cambiaría para obtener el punto de saturación en un número mayor de usuarios.**

Visa Pool used Conns	HTTP Current Threads Busy	Conn queued, instant
0.00	1.86	1.57
0.00	5.00	505.81
0.00	5.00	518.94
0.00	5.00	508.75
0.10	5.00	511.90
0.14	5.00	517.86
0.36	5.00	545.79
0.10	4.60	542.30
0.00	5.00	524.67
nan	nan	nan

Podemos apreciar como el parámetro que más sufre es el número de hilos en uso, ya que tenemos un máximo asignado de 5. Por tanto, seguramente si aumentamos este número consigamos un mejor rendimiento.

- **Realizar el ajuste correspondiente en el servidor de aplicaciones, reiniciarlo y tomar una nueva muestra cercana al punto de saturación. ¿Ha mejorado el rendimiento del sistema? Documente en la memoria de prácticas el cambio realizado y la mejora obtenida**



Tras poner un número mínimo de 10 y un máximo de 20, obtenemos que, con 8000 usuarios. tenemos un *throughput* de 370/s en el total y de 185.82/s en *procesapago*. Con lo cual, tenemos una mejora del rendimiento. Además, gracias a

la monitorización, vemos que usamos una media de 13.5 hilos, con lo que tenemos el sistema algo más liberado.

(Estos resultados se pueden comprobar en la carpeta *Adjuntos de ejercicio 9*).