		Escuela Politécnica Superior Ingeniería Informática Prácticas de Sistemas Informáticos 2			
Grupo	2321	Práctica	2	Fecha	26/04/2021
Alumno/a		Arribas Gozalo, Francisco Javier			
Alumno/a		Saenz Ferrero, Santos			

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.

Se definió el plan de pruebas que se adjunta sin mayores problemas.

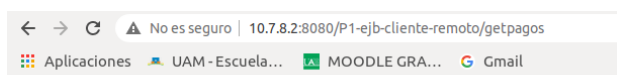
Ejercicio 2:

Preparar el PC con el esquema descrito en la Figura 22. Para ello:

- Anote en la memoria de prácticas las direcciones IP asignadas a las máquinas virtuales y al PC.
Las direcciones asignadas a las dos maquinas virtuales fueron respectivamente 10.7.8.1 y 10.7.8.2, mientras que al pc host fue 10.7.8.9
- Detenga el servidor de GlassFish del PC host.
- Inicie los servidores GlassFish en las máquinas virtuales.
- Repliegue todas las aplicaciones o pruebas anteriores (P1-base, P1-ws, etc), para limpiar posibles versiones incorrectas.
- Revise y modifique si es necesario los ficheros build.properties (propiedad “nombre”) de cada versión, de modo que todas las versiones tengan como URL de despliegue las anteriormente indicadas.
- Revise y modifique si es necesario el fichero glassfish-web.xml, para indicar la IP del EJB remoto que usa P1-ejb-cliente.
- Despliegue las siguientes prácticas: P1-base, P1-ws, P1-ejb-servidor-remoto y P1-ejb-cliente-remoto, con el siguiente esquema:
 - El destino de despliegue de la aplicación P1-base será PC2VM con IP10.X.Y.2 (as.host)
 - El destino del despliegue de la parte cliente de P1-ws y de P1-ejb-cliente-remoto será PC2VM con IP 10.X.Y.2 (as.host.client de P1-ws y as.host de P1-ejb-cliente-remoto)
 - El destino del despliegue de la parte servidor de P1-ws y de P1-ejb-servidor-remoto será PC1VM con IP 10.X.Y.1 (as.host.server de P1-ws y as.host.server y as.host.client de P1-ejb-servidor-remoto)
 - La base de datos en todos ellos será la de PC1VM con IP 10.X.Y.1(db.host)

Tras detener/iniciar todos los elementos indicados, anotar la salida del comando “free” así como un pantallazo del comando “nmon” (pulsaremos la tecla “m” para obtener el estado de la RAM) tanto en las máquinas virtuales como en el PC host. Anote sus comentarios en la memoria. Pruebe a ejecutar un pago “de calentamiento” por cada uno de los métodos anteriores y verifique que funciona a través de la página testbd.jsp.

Comprobacion de pagos correcta verificada a partir de las siguientes capturas



Pago con tarjeta

Lista de pagos del comercio 1

idTransaccion	Importe	codRespuesta	idAutorizacion
1	10.0	000	2

[Volver al comercio](#)



Pago con tarjeta

Lista de pagos del comercio 1

idTransaccion	Importe	codRespuesta	idAutorizacion
1	10.0	000	3

[Volver al comercio](#)

Una vez ejecutado el comando free obtenemos una tabla donde se muestra que se dispone para el pc host de una memoria RAM de tamaño 4002268 Bytes, de los cuales están siendo usados 3196544 Bytes y están libres 107476 Bytes. Por otra parte se indica que comparte con el buffer 148888 Bytes. La cache cuenta con 698248 Bytes y de esto tiene disponible 418828 Bytes. Para el disco duro tenemos un espacio de 7999484 Bytes, de los cuales 334848 Bytes están siendo usados y 7664636 Bytes están libres

Para la mv1 obtenemos una memoria RAM de tamaño 767168 Bytes, de los cuales están siendo usados 617672 Bytes y están libres 149496 Bytes. Por otra parte se indica que comparte con el buffer y con cache 0 Bytes. Los buffers cuentan con un espacio de 33412 Bytes y están cacheados 185256 Bytes. Por último se indica que el disco duro cuenta con un espacio de 153592 Bytes de los cuales no se está usando ninguno, y están por lo tanto todos ellos libres

Para la mv2 obtenemos una memoria RAM de tamaño 767168 Bytes, de los cuales están siendo usados 680772 Bytes y están libres 86396 Bytes. Por otra parte se indica que comparte con el buffer y con cache 0 Bytes. Los buffers cuentan con un espacio de 20252 Bytes y están cacheados 179228 Bytes. Por último se indica que el disco duro cuenta con un espacio de 153592 Bytes de los cuales no se está usando ninguno, y están por lo tanto todos ellos libres

A partir de las siguientes imágenes se comprueba la salida del nmon en pc host y ambas maquinas virtuales.

```

nmon-16g-----Hostname=labvirtaps---Refresh= 2secs ---07:28.20---
Memory and Swap
PageSize:4KB  RAM-Memory  Swap-Space      High-Memory  Low-Memory
Total (MB)      3908.5      7812.0      - not in use - not in use
Free (MB)       132.5      7740.2
Free Percent     3.4%      99.1%
Linux Kernel Internal Memory (MB)
                        Cached=      771.2      Active=      2034.8
Buffers=      115.5 Swapcached=      11.4 Inactive =      849.0
Dirty  =       0.2 Writeback =       0.0 Mapped  =      516.9
Slab   =      199.7 Commit_AS =     8734.6 PageTables=      69.2

```

Salida comando nmon pc host.

```

nmon-12f-----[H for help]---Hostname=si2srv01---Refresh= 2secs ---23:32.16---
Memory Stats
      RAM      High      Low      Swap
Total MB      749.2      0.0      749.2      150.0
Free MB       144.8      0.0      144.8      150.0
Free Percent   19.3%      0.0%      19.3%      100.0%
      MB
      Cached=      181.0      Active=      450.3
Buffers=      32.9 Swapcached=      0.0 Inactive =      130.4
Dirty  =       0.0 Writeback =      0.0 Mapped  =      30.2
Slab   =      14.1 Commit_AS =     1157.2 PageTables=      2.4

```

Salida comando nmon mv1.

```

nmon-12f-----[H for help]---Hostname=si2srv02---Refresh= 2secs ---23:32.18---
Memory Stats
      RAM      High      Low      Swap
Total MB      749.2      0.0      749.2      150.0
Free MB       82.4      0.0      82.4      150.0
Free Percent   11.0%      0.0%      11.0%      100.0%
      MB
      Cached=      175.1      Active=      528.5
Buffers=      20.3 Swapcached=      0.0 Inactive =      117.0
Dirty  =       0.0 Writeback =      0.0 Mapped  =      22.5
Slab   =      13.0 Commit_AS =     1199.9 PageTables=      1.5

```

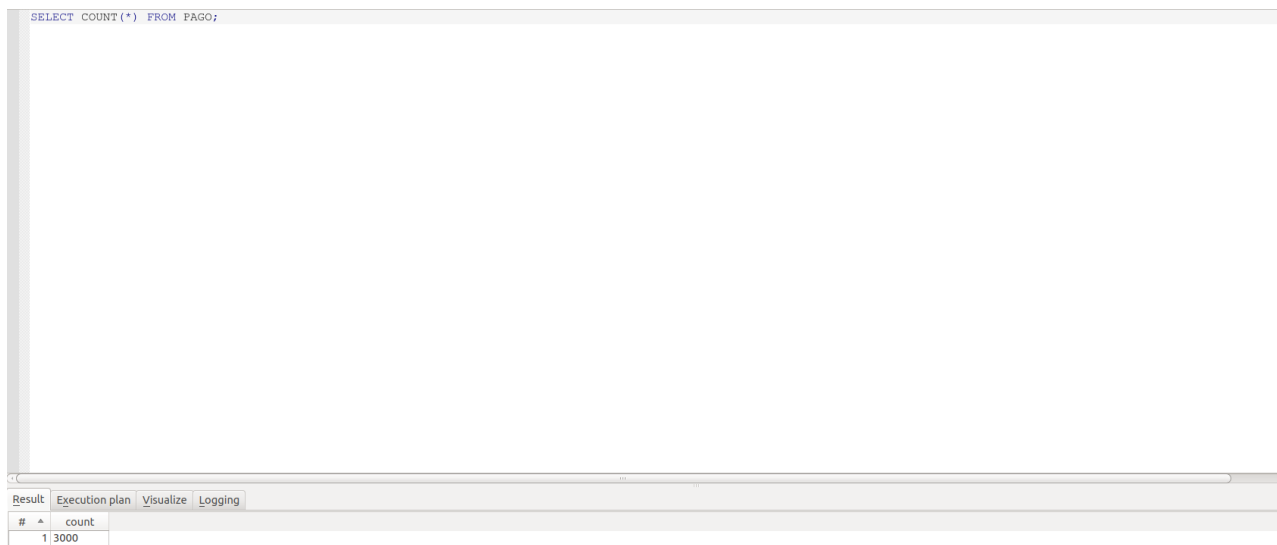
Salida comando nmon mv2.

Ejercicio 3:

Ejecute el plan completo de pruebas sobre las 3 versiones de la práctica, empleando el esquema de despliegue descrito anteriormente. Realice la prueba tantas veces como necesite para eliminar ruido relacionado con procesos periódicos del sistema operativo, lentitud de la red u otros elementos.

- Compruebe que efectivamente se han realizado todos los pagos. Es decir, la siguiente consulta deberá devolver “3000”:

```
SELECT COUNT (*) FROM PAGO;
```



#	count
1	3000

Compruebe que ninguna de las peticiones ha producido un error. Para ello revise que la columna %Error indique 0% en todos los casos.

Una vez que los resultados han sido satisfactorios:

- Anote los resultados del informe agregado en la memoria de la práctica.

Los resultados del informe agregado son los siguientes

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimie...	Kb/sec	Sent KB/sec
P1-base	1000	7	6	8	9	13	5	530	0,00%	130,8/sec	167,44	0,00
P1-ws	1000	47	42	58	67	106	37	721	0,00%	21,1/sec	27,31	0,00
P1-ejb	1000	12	11	16	17	23	8	450	0,00%	80,6/sec	104,90	0,00
Total	3000	22	11	46	53	79	5	721	0,00%	44,5/sec	57,48	0,00

- Salve el fichero server.log que se encuentra en la ruta glassfish/domains/domain1/logs de Glassfish y adjúntelo con la práctica.
- Añada a la memoria de prácticas la siguiente información: ¿Cuál de los resultados le parece el mejor? ¿Por qué? ¿Qué columna o columnas elegiría para decidir este resultado?

El que parece mejor resultado es P1-base ya que es la que más peticiones administra por segundo y la que menos tiempo tarda en procesar estas. Esto se debe a que a que tiene un rendimiento de 130.8 peticiones por segundo, además de una media de 7 ms por petición.

Por otra parte también es relevante la columna 90% line que indica que también es este servidor el que procesa la mayoría de peticiones, el 90%, mas rápido.

Incluir el directorio P2 en la entrega.

Repita la prueba de P1-ejb (inhabilítelos “ThreadGroup” P1-base y P1-ws) con el EJB local incluido en P1-ejb-servidor-remoto. Para ello, cambie su “HTTPRequest”, estableciendo su “ServerName or IP” a 10.X.Y.1 (VM1) y su “Path” a “P1-ejb-cliente/procesapago”. Compare los resultados obtenidos con los anteriores.

El fichero P2.jmx entregado no debe contener estos cambios, es decir, debe estar configurado para probar el EJB remoto.

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Min	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec
P1-obj	1000	4	5	6	7	11	3	85	0,00%	188,1/sec	144,19	0,00
Total	1000	4	5	6	7	11	3	85	0,00%	188,1/sec	144,19	0,00

Esta aplicación ahora a mejorar notablemente su rendimiento pasando de 80.6 pet/seg a 188.1 pet/seg además pasa de una media de 12ms a 4ms, es decir de media tarda un tercio en procesar las peticiones respecto a antes. El 90% de las peticiones por otra parte pasan de procesarse de 16 ms a 6ms. Estas y las otras mejoras se deben debido a que al cambiar la IP de la request a 10.7.8.1 se va a ejecutar de forma local eliminandose la problemática de los retardos en las comunicaciones.

Ejercicio 4:

Adaptar la configuración del servidor de aplicaciones a los valores indicados. Guardar, como referencia, la configuración resultante, contenida en el archivo de configuración localizado en la máquina virtual en \$opt/glassfish4/glassfish/domains/domain1/config/domain.xml. Para obtener la versión correcta de este archivo es necesario detener el servidor de aplicaciones. Incluir este fichero en el entregable de la práctica. Se puede copiar al PC con scp.

Revisar el script si2-monitor.sh e indicar los mandatos asadmin que debemos ejecutar en el PC host para averiguar los valores siguientes, mencionados en el Apéndice 1, del servidor PC1VM1:

En la carpeta Evidencias/e4 se encuentra el fichero domain.xml con los cambios realizados.

1. Max Queue Size del Servicio HTTP

Debemos ejecutar el comando:

```
asadmin --host 10.7.8.1 -user admin -passwordfile ./passwordfile get
server.thread-pools.thread-pool.http-thread-pool.maxqueue-size
```

El valor esperado de respuesta es el el tamaño máximo de la cola, en este caso, por defecto, 4096.

2. Maximum Pool SizedelPool de conexiones a nuestra DB

Debemos ejecutar el comando:

```
asadmin --host 10.7.8.1 -user admin -passwordfile ./passwordfile get
resources.jdbc-connection-pool.VisaPool.max-pool-size
```

El valor esperado de respuesta es el el tamaño máximo de hilos que pueden conectarse a nuestra base de datos, en este caso, por defecto, 32.

Así como el mandato para monitorizar el número de errores en las peticiones al servidor web.

Debemos ejecutar el comando:

```
asadmin --host 10.7.8.1 -user admin -passwordfile ./passwordfile monitor -type
httplistener server
```

El valor esperado de respuesta es una tabla, entre las cuales se muestra la columna ec, o error count.

Ejercicio 5:

Registrar en la hoja de cálculo de resultados los valores de configuración que tienen estos parámetros.

En el fichero adjunto SI2-P2-curvaProductividad.ods se han incluido los valores de configuración de los parametros indicados (-Xmx, -Xms, Max Thread Pool Size, Max Queue Size, Max Sessions y Maximum Pool Size).

Elemento	Parámetro	Valor
JVM Settings	Heap Máx. (MB)	512
JVM Settings	Heap Mín. (MB)	512
HTTP Service	Max.Thread Count	5
HTTP Service	Queue size	4096
Web Container	Max.Sessions	-1
Visa Pool	Max.Pool Size	32

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:

- 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...)

Al usar una CPU potente, y solo tener un thread (cliente) haciendo peticiones secuenciales, en nuestro caso, el elemento más costoso del proceso es el disco. Con más clientes, sin embargo, podemos imaginar que la CPU empezará a ser un elemento más costoso y difícil de mejorar.

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

No se trata de una simulación particularmente realista, ya que un solo cliente haciendo peticiones de forma secuencial no es una situación normal para un servidor.

Además, al estar cliente y servidor en la misma máquina, la red es más rápida de lo que sería normalmente.

Por ello, podemos llegar a alcanzar conclusiones erróneas.

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

Para mejorar el rendimiento de escritura y lectura en disco, podríamos usar un disco SSD o M2, que tiene una velocidad de lectura/escritura más rápida. También se podría implementar un caché o un servicio de sesiones en el servidor, que aliviaría la cantidad de consultas al disco cuando hay varias peticiones de un mismo cliente.

Si, por otro lado, nos preocupa como sería el escalado del servidor con nuestra CPU actual, deberíamos, o bien mejorar la CPU, o bien emplear más CPUs, o, como solución más barata, usar el sistema EJB, donde el servidor no es responsable de toda la computación en una transacción.

Nota: Las respuestas a estas cuestiones deben estar acompañadas por datos que respalden la argumentación, en forma de pantallazos de nmon (o gráficas de Nmon Visualizer) y pantallazos de si2-monitor.sh.

En la carpeta Evidencias/e6 se han incluido los ficheros nmon de ambas máquinas, así como el resultado de ejecutar si2-monitor.sh y todas las capturas de los datos con NMON Visualizer.

Ejercicio 7:

Preparar el script de Jmeter para su ejecución en el entorno de pruebas. Cambiar la dirección destino del servidor para que acceda al host en el que se encuentra el servidor de aplicaciones. Crear también el directorio datagen en el mismo directorio donde se encuentre el script, y copiar en él el archivo listado.csv, ya que, de dicho archivo, al igual que en los ejercicios anteriores, se obtienen los datos necesarios para simular el pago.

A continuación, realizar una ejecución del plan de pruebas con un único usuario, una única ejecución, y un *think time* bajo (entre 1 y 2 segundos) para verificar que el sistema funciona correctamente. Comprobar, mediante el *listener View Results Tree* que las peticiones se ejecutan correctamente, no se produce ningún tipo de error y los resultados que se obtienen son los adecuados.

Una vez comprobado que todo el proceso funciona correctamente, desactivar dicho *listener* del plan de pruebas para que no aumente la carga de proceso de Jmeter durante el resto de la prueba. Este ejercicio no genera información en la memoria de la práctica, realícelo únicamente para garantizar que la siguiente prueba va a funcionar.

Si bien no se ha pedido, en la carpeta Evidencias/e7 se ha incluido una captura mostrando el correcto funcionamiento de la prueba, mostrando el árbol de resultados.

Ejercicio 8:

Obtener la curva de productividad, siguiendo los pasos que se detallan a continuación:

- Previamente a la ejecución de la prueba se lanzará una ejecución del *script* de pruebas (unas 10 ejecuciones de un único usuario) de la que no se tomarán resultados, para iniciar el sistema y preparar medidas consistentes a lo largo de todo el proceso.
Borrar los resultados de la ejecución anterior. En la barra de acción de Jmeter, seleccionar Run→ClearAll.
- Borrar los datos de pagos en la base de datos VISA.
- Ejecutar la herramienta de monitorización nmon en ambas máquinas, preferiblemente en modo "Data-collect"(Ver8.2.2).
- Seleccionar el número de usuarios para la prueba en Jmeter (parámetro C de la prueba)
- Conmutar en Jmeter a la pantalla de presentación de resultados, *AggregateReport*.
- Ejecutar la prueba. En la barra de acción de Jmeter, seleccionar Run→Start.
- Ejecutar el programa de monitorización si2-monitor.sh
 - Arrancarlo cuando haya pasado el tiempo definido como rampa de subida de usuarios en Jmeter (el tiempo de ejecución en Jmeter se puede ver en la esquina superior derecha de la pantalla).
 - Detenerlo cuando esté a punto de terminar la ejecución de la prueba. Este momento se puede detectar observando cuando el número de hilos concurrentes en Jmeter (visible en la esquina superior derecha) comienza a disminuir (su máximo valor es C).
 - Registrar los resultados que proporciona la monitorización en la hoja de cálculo.
- Durante el periodo de monitorización anterior, vigilar que los recursos del servidor si2srv02 y del ordenador que se emplea para realizar la prueba no se saturen. En caso de usar nmon de forma interactiva, se deben tomar varios pantallazos del estado de la CPU durante la prueba, para volcar en la hoja de cálculo del dato de uso medio de la CPU (CPUaverage%). En caso de usar nmon en modo "Data-collect", esta información se puede ver posteriormente en NmonVisualizer. Una tercera opción (recomendada) es ejecutar el comando vmstat en una terminal remota a la máquina si2srv02, para extraer directamente el valor de uso medio de su CPU5.
- Finalizada la prueba, salvar el resultado de la ejecución del Aggregate Report en un archivo, y registrar en la hoja de cálculo de resultados los valores Average, 90% line y Throughput para las siguientes peticiones:
 - ProcesaPago.
 - Total.

Una vez realizadas las iteraciones necesarias para alcanzar la saturación, representar la curva de Throughput versus usuarios. Incluir el fichero P2-curvaProductividad.jmx en la entrega.

Nota: Todos los datos de monitorización que se entreguen como parte de la curva de rendimiento (derivados de Jmeter, nmon y si2-monitor.sh) deben estar respaldados por pruebas que demuestren su veracidad, ya sea en la propia memoria o en ficheros adicionales (recomendado). En el caso de los recursos del sistema, se pueden mostrar tanto pantallazos del modo interactivo de nmon como gráficas de NMONvisualizer.

En el fichero adjunto SI2-P2-curvaProductividad.ods se han incluido los resultados de todas las pruebas pedidas, además de las curvas pedidas.

En la carpeta Evidencias/e8 se encuentran todas las evidencias necesarias: el resultado de la monitorización CPU con vmstat, el resultado del script de monitorización y la tabla de resultados agregados para cada una de las pruebas realizadas.

Si bien en el enunciado se sugirió realizar pruebas con incrementos de 250 usuarios, al emplear una CPU considerablemente más potente a la de los laboratorios, hemos necesitado hacer incrementos mayores para apreciar los resultados de la saturación del servidor. Así, se incluyen evidencias de pruebas para 1, 1000, 3000, 5000, 5500, 5750, 5950, 6000, 6050, 6250, 6500, 7000 y 10000 clientes.

Ejercicio 9:

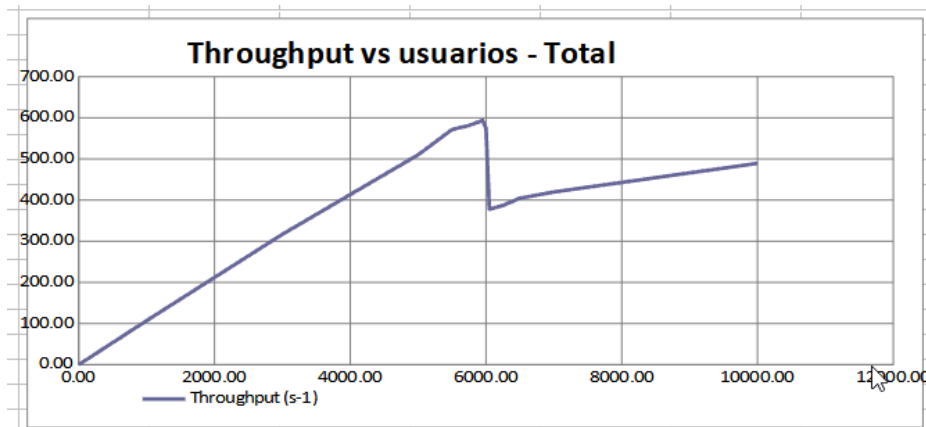
Responda a las siguientes cuestiones:

En las carpetas Evidencias/e8 y Evidencias/e9 se encuentran todas las pruebas que respaldan los datos ofrecidos en este ejercicio. En el fichero adjunto SI2-P2-curvaProductividad.ods podremos apreciar las curvas de productividad de las que se habla.

- **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.**

Para calcular el punto de saturación, se calcula el punto exacto en el que se cruzan las rectas de la denominada “zona lineal” y la línea de saturación.

Observando la siguiente tabla:



Podemos ver que la zona lineal llega hasta los 6000 usuarios. Con un throughput de 600 para 6000 usuarios, definimos la recta como:

- **$Y = 0.1X$**

Por otro lado, la línea de saturación es una línea paralela al eje X que pase por el valor más alto de throughput: en nuestro caso, este valor es 593.7, y por tanto, la línea de saturación es:

- **$Y = 593.7$**

El punto de intersección de ambas rectas es **(5937, 593.7)**. Con esto, podemos afirmar que **el sistema no está diseñado para superar 5937 usuarios en ningún momento determinado**, y deberíamos implementarlo teniendo esto en cuenta.

En la zona de saturación, observamos que con 5950 clientes se produce un throughput de 593.7, y es el máximo valor que se obtiene en la zona de saturación.

- **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.**

A pesar del grande número de clientes, la CPU no ha llegado a saturarse completamente. Así, determinamos que el mayor factor es el número de Thread Pool que el servidor procesa al mismo tiempo. En este caso, nuestro valor de Thread Pool era 5, y esto significa que el servidor solo era capaz de procesar 5 peticiones al mismo tiempo. Aumentando este número, podemos también aumentar el número mayor de usuarios soportados sin saturar el servidor.

- **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.**

En la carpeta Evidencias/e9 se encuentra la evidencia de esta última prueba. Se ha aumentado el número de Thread Pool de 5 máximo a 10 máximo. Así, se pueden llegar a procesar el doble de peticiones al mismo tiempo, si el servidor lo permite. Se ha probado una muestra cercana, pero superior, al punto de saturación – 6500 clientes. En las pruebas adjuntas, se puede observar que el rendimiento ha aumentado con respecto a la configuración anterior, con un throughput de 504.84 comparado al anterior de 404.6 para los mismos clientes.