		Escuela Politécnica Superior Ingeniería Informática Prácticas de Sistemas Informáticos 2			
Grupo	2362	Práctica	2	Fecha	14/04/2020
Alumno/a		Ramos Pedroviejo, Alba			
Alumno/a		Serrano Salas, Nicolás			

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.

Importante: Para comprobar el correcto funcionamiento de la simulación y detectar posibles fallos, se recomienda añadir también al elemento P2 Test un “árbol de resultados” (View Results Tree). Para ello, sobre el plan de pruebas, botón derecho, Add Listener View Results Tree. Una vez se tenga la certeza de que la simulación funciona correctamente se desactivará el “árbol de resultados” (pulsando encima con el botón derecho del ratón) y se realizará de nuevo la simulación. El árbol de resultados permite inspeccionar los datos enviados en cada petición HTTP y la respuesta obtenida del servidor, que deberán ser correctas. Por ejemplo, no deberá aparecer ningún pago incorrecto en las respuestas.

Se ha creado el fichero P2.jmx siguiendo las instrucciones indicadas.

Ejercicio 2:

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

- Anote en la memoria de prácticas las direcciones IP asignadas a cada PC.
PC1VM tiene la IP 10.8.6.1 y PC2VM tiene la IP 10.8.6.2
- Detenga el servidor de GlassFish de los PCs físicos
- 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-jeb-cliente-remoto, con el siguiente esquema:
 - El destino de despliegue de la aplicación P1-base será PC2VM con IP 10.X.Y.2 (as.host)

Hemos cambiado el fichero build.properties donde teníamos as.host 10.8.6.1 por 10.8.6.1

- 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)

Mientras que los datos del fichero build.properties de P1-ws se mantienen constantes, en P1-ejb-cliente-remoto hemos cambiado as.host que teníamos 10.8.6.1 por la dirección 10.8.6.2

- 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)

Mientras que los datos del fichero build.properties de P1-ws se mantienen constantes, en P1-ejb-servidor-remoto hemos cambiado as.host.server y as.host.client de la dirección 10.8.6.2 a la 10.8.6.1

- La base de datos en todos ellos será la de PC1VM con IP 10.X.Y.1 (db.host)

También hemos revisado todos y cada uno de los ficheros postgresql.properties y db.host se ha creado correctamente.

Pero en P1-base hemos tenido que cambiar db.client.host de 10.8.6.1 a 10.8.6.2 y en P1-ejb-servidor-remoto hemos tenido que cambiar db.client.host de 10.8.6.2 a 10.8.6.1.

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 los PCs físicos. Anote sus comentarios en la memoria.

Hemos ejecutado la práctica usando un solo ordenador debido a las circunstancias excepcionales en las que nos encontramos. Los comandos muestran cuánta RAM tiene asignada tanto el pc como las máquinas virtuales. Muestran cuánta memoria hay libre y en uso, así como qué uso se le está dando. El comando nmon es más detallado y más visual que el comando free.

En pc:

```
alba@alba-Aspire-E5-574:~$ free
              total usado libre compartido búfer/caché disponible
Memoria:    3902684 2018392 140644 1315220 1743648 3330988
Swap:       2097148 1853416 243732
```

```
nmon-16g-----Hostname=alba-Aspire-ERefresh= 2secs ---19:04.
Memory and Swap
PageSize:4KB  RAM-Memory  Swap-Space  High-Memory  Low-Memory
Total (MB)    3811.2      2048.0      - not in use - not in use
Free (MB)     167.7      223.9
Free Percent   4.4%         10.9%
Linux Kernel Internal Memory (MB)
                Cached=    1632.2    Active=    2546.6
Buffers=       27.6  Swapcached=    5.4  Inactive =    761.2
Dirty  =        1.2  Writeback =    0.0  Mapped   =    848.0
Slab   =       142.8  Commit_AS =  10713.3  PageTables=    69.7
```

En máquina virtual 1:

```
si2@si2srv01:~$ free
              total        used        free      shared    buffers     cached
Mem:         767168      557152      210016           0       29340     184232
-/+ buffers/cache:      343580      423588
Swap:        153592           0       153592
```

```
mmon-12f-----Hostname=si2srv01-----Refresh= 2secs ---11:03.20---
Memory Stats
RAM      High      Low      Swap
Total MB 749.2    0.0     749.2   150.0
Free MB   197.9    0.0     197.9   150.0
Free Percent 26.4%  0.0%    26.4%  100.0%
MB
Cached= 185.1  Active= 395.7
Buffers= 29.0 Swapcached= 0.0 Inactive = 133.2
Dirty = 0.2 Writeback = 0.0 Mapped = 26.5
Slab = 13.9 Commit_AS = 1066.7 PageTables= 1.6
```

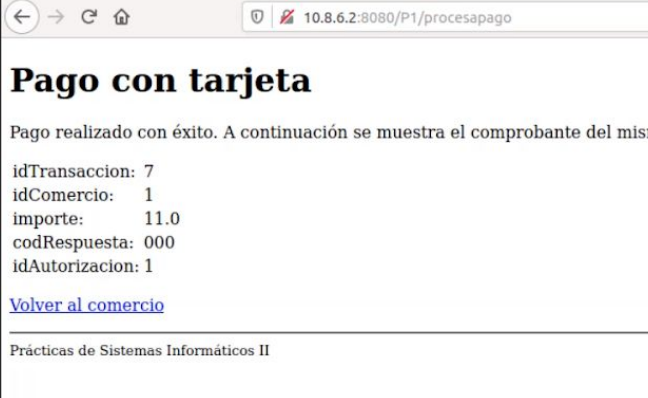
En máquina virtual 2:

```
si2@si2srv02:~$ free
              total        used        free      shared    buffers     cached
Mem:         767168      536240      230928           0       18428     266592
-/+ buffers/cache:      251220      515948
Swap:        153592           0       153592
```

```
mmon-12f-----Hostname=si2srv02-----Refresh= 2secs ---11:02.01---
Memory Stats
RAM      High      Low      Swap
Total MB 749.2    0.0     749.2   150.0
Free MB   224.0    0.0     224.0   150.0
Free Percent 29.9%  0.0%    29.9%  100.0%
MB
Cached= 260.5  Active= 290.9
Buffers= 18.1 Swapcached= 0.0 Inactive = 212.0
Dirty = 0.0 Writeback = 0.0 Mapped = 22.7
Slab = 14.3 Commit_AS = 915.0 PageTables= 1.3
```

Pruebe a ejecutar un pago “de calentamiento” por cada uno de los métodos anteriores y verifique que funciona través de la página testbd.jsp.

P1-base



The screenshot shows a web browser window with the address bar displaying '10.8.6.2:8080/P1/procesapago'. The page title is 'Pago con tarjeta'. The main content area displays a confirmation message: 'Pago realizado con éxito. A continuación se muestra el comprobante del mismo'. Below this, the transaction details are listed: 'idTransaccion: 7', 'idComercio: 1', 'importe: 11.0', 'codRespuesta: 000', and 'idAutorizacion: 1'. A blue link 'Volver al comercio' is provided. The footer contains the text 'Prácticas de Sistemas Informáticos II'.

Pago con tarjeta

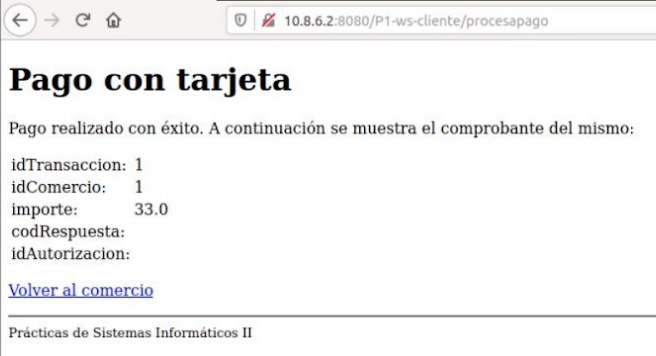
Pago realizado con éxito. A continuación se muestra el comprobante del mismo

idTransaccion: 7
idComercio: 1
importe: 11.0
codRespuesta: 000
idAutorizacion: 1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

P1-ws



The screenshot shows a web browser window with the address bar displaying '10.8.6.2:8080/P1-ws-cliente/procesapago'. The page title is 'Pago con tarjeta'. The main content area displays a confirmation message: 'Pago realizado con éxito. A continuación se muestra el comprobante del mismo:'. Below this, the transaction details are listed: 'idTransaccion: 1', 'idComercio: 1', 'importe: 33.0', 'codRespuesta:', and 'idAutorizacion:'. A blue link 'Volver al comercio' is provided. The footer contains the text 'Prácticas de Sistemas Informáticos II'.

Pago con tarjeta

Pago realizado con éxito. A continuación se muestra el comprobante del mismo:

idTransaccion: 1
idComercio: 1
importe: 33.0
codRespuesta:
idAutorizacion:

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

P1-ejb



The screenshot shows a web browser window with the address bar displaying '10.8.6.1:8080/P1-ejb-cliente/procesapago'. The page title is 'Pago con tarjeta'. The main content area displays a confirmation message: 'Pago realizado con éxito. A continuación se muestra el comprobante del mismo:'. Below this, the transaction details are listed: 'idTransaccion: 9', 'idComercio: 1', 'importe: 41.0', 'codRespuesta: 000', and 'idAutorizacion: 3'. A blue link 'Volver al comercio' is provided. The footer contains the text 'Prácticas de Sistemas Informáticos II'.

Pago con tarjeta

Pago realizado con éxito. A continuación se muestra el comprobante del mismo:

idTransaccion: 9
idComercio: 1
importe: 41.0
codRespuesta: 000
idAutorizacion: 3

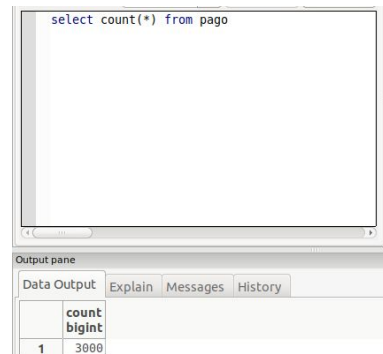
[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

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;



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

En la imagen del reporte adjunta más abajo podemos comprobar que no ha habido ningún error.

Una vez que los resultados han sido satisfactorios:

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

Incluir el directorio P2 en la entrega.

Tras realizar el plan de pruebas, obtenemos los siguientes resultados en el informe agregado:

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Min	Max	% Error	Rendimiento	Kb/sec	Sent KB/sec
l-base	1000	14	12	25	38	47	6	55	0,00%	65,9/sec	49,83	0,00
l-ws	1000	98	83	145	164	214	49	957	0,00%	10,2/sec	7,74	0,00
l-ejb	1000	34	30	59	63	70	12	81	0,00%	28,7/sec	22,12	0,00
Total	3000	49	32	116	136	174	6	957	0,00%	20,2/sec	15,41	0,00

A la vista de los resultados, parece que las columnas “rendimiento” y “Media” son las más adecuadas para decidir cuál de los 3 resultados es el mejor. Según estas columnas, parece que P1-base es el que mejores resultados obtiene, ya que tiene mayor rendimiento y menos tiempo por petición de media al ejecutar las 1000 pruebas.

Repita la prueba de P1-ejb (inhabilite los ‘Thread Group’ P1-base y P1-ws) con el EJB local incluido en P1- ejb-servidor-remoto. Para ello, cambie su „HTTP Request’, estableciendo su ‘Server Name or IP’ a 10.X.Y.1 (VM1) y su ‘Path’ a ‘P1-ejb-cliente/procesapago’. Compare los resultados obtenidos con los anteriores.

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Min	Máx.	% Error	Rendimie...	Kb/sec	Sent KB/s...
P1-ejb	1000	19	11	41	44	49	4	3789	0,00%	50,0/sec	38,15	0,00
Total	1000	19	11	41	44	49	4	3789	0,00%	50,0/sec	38,15	0,00

Se puede observar que los resultados han mejorado significativamente respecto a usar el cliente remoto. Esto es lógico, ya que usando el cliente local no tendremos que conectarnos con el cliente remoto y ahorraremos ese tiempo.

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 del laboratorio con scp.

Primero hemos eliminado -client y hemos añadido -server

Select	Value
<input type="checkbox"/>	-Dorg.glassfish.additionalOSGiBundlesToStart=org.apache.felix.shell,org.apache.felix.gogo.runtime,org.apache.felix.gogo.shell,org.apache.felix.gogo.commands
<input type="checkbox"/>	-Djavax.management.builder.initial=com.sun.enterprise.v3.admin.AppServerMBeanServerBuilder
<input type="checkbox"/>	-Djavax.net.ssl.keyStore=\${com.sun.aas.instanceRoot}/config/keystore.jks
<input type="checkbox"/>	-server
<input type="checkbox"/>	-DANTLR_USE_DIRECT_CLASS_LOADING=true
<input type="checkbox"/>	-Dcom.ctc.wstx.returnNullForDefaultNamespace=true
<input type="checkbox"/>	-Dfelix.fileinstall.bundles.startTransient=true
<input type="checkbox"/>	-Djavax.net.ssl.trustStore=\${com.sun.aas.instanceRoot}/config/cacerts.jks
<input type="checkbox"/>	-Dosgi.shell.telnet.ip=127.0.0.1
<input type="checkbox"/>	-Dfelix.fileinstall.log.level=2
<input type="checkbox"/>	-XX:+UnlockDiagnosticVMOptions
<input type="checkbox"/>	-Djava.security.auth.login.config=\${com.sun.aas.instanceRoot}/config/login.conf
<input type="checkbox"/>	-Dfelix.fileinstall.disableConfigSave=false
<input type="checkbox"/>	-Djava.awt.headless=true
<input type="checkbox"/>	-Xmx512m

Luego añadimos el Heap mínimo y hemos comprobado que el Heap máximo es 512

<input type="checkbox"/>	-Djava.awt.headless=true
<input type="checkbox"/>	-Xmx512m
<input type="checkbox"/>	-Djdbc.drivers=org.apache.derby.jdbc.ClientDriver
<input type="checkbox"/>	-Djdk.corba.allowOutputStreamSubclass=true
<input type="checkbox"/>	-Dosgi.shell.telnet.port=6666
<input type="checkbox"/>	-Dosgi.shell.telnet.maxconn=1
<input type="checkbox"/>	-Djavax.xml.accessExternalSchema=all
<input type="checkbox"/>	-Djava.ext.dirs=\${com.sun.aas.javaRoot}/lib/ext\${path.separator}\${com.sun.aas.javaRoot}/re/lib/ext\${path.separator}\${com.sun.aas.instanceRoot}/lib/ext
<input type="checkbox"/>	-Djava.security.policy=\${com.sun.aas.instanceRoot}/config/server.policy
<input type="checkbox"/>	-Dgosh.args=--nointeractive
<input type="checkbox"/>	-Xms512m
<input type="checkbox"/>	-Dcom.sun.enterprise.config.config_environment_factory_class=com.sun.enterprise.config.serverbeans.AppserverConfigEnvironmentFactory
<input type="checkbox"/>	-XX:MaxPermSize=192m

Tras esto hemos eliminado el checkbox de Reload y Auto Deploy y seleccionado la checkbox de Precompile.

Reload: ☐ **Enabled**
Enables dynamic reloading of applications.

Reload Poll Interval: **Seconds**
Frequency for checking reload requests.

Admin Session Timeout: **Minutes**
A value of 0 means the session never times out.

Auto Deploy Settings

Auto Deploy: ☐ **Enabled**
Automatically deploys applications in the autodeploy directory.

Auto Deploy Poll Interval: **Seconds**
Frequency at which the autodeploy directory is checked for applications; interval does not affect amount of time to load the application or module.

Auto Deploy Retry Timeout: **Seconds**
Time to report failure after a file remains stable in size but cannot be opened.

Auto Deploy Directory:
Directory to monitor for autodeploy applications.

XML Validation:
Type of deployment descriptor validation.

Verifier: ☐ **Enabled**
Performs detailed verification before deployment.

Precompile: ☒ **Enabled**
Precompiles JSPs, deploys only resulting class files.

Finalmente hemos puesto la monitorización en nivel alto para los servicios indicados, y puesto en off el resto.

Component Level Settings (16)		
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Level: <input type="text" value="Change Level"/>
Select	Module	Monitoring Level
<input type="checkbox"/>	Jvm	HIGH
<input type="checkbox"/>	Transaction Service	OFF
<input type="checkbox"/>	Connector Service	OFF
<input type="checkbox"/>	Jms Service	OFF
<input type="checkbox"/>	Security	OFF
<input type="checkbox"/>	Web Container	HIGH
<input type="checkbox"/>	Jersey(RESTful Web Services)	OFF
<input type="checkbox"/>	Web Services Container	OFF
<input type="checkbox"/>	Java Persistence	OFF
<input type="checkbox"/>	Jdbc Connection Pool	HIGH
<input type="checkbox"/>	Thread Pool	HIGH
<input type="checkbox"/>	Ejb Container	OFF
<input type="checkbox"/>	ORB (Object Request Broker)	OFF
<input type="checkbox"/>	Connector Connection Pool	OFF
<input type="checkbox"/>	Deployment	OFF
<input type="checkbox"/>	Http Service	HIGH

Revisar el script si2-monitor.sh e indicar los mandatos asadmin que debemos ejecutar en el Host PC1 para averiguar los valores siguientes, mencionados en el Apéndice 1, del servidor PC1VM1:

1. Max Queue Size del Servicio HTTP

Hemos ejecutado el siguiente comando, obteniendo un tamaño 4096:

```
alba@alba-Aspire-E5-574:~/Escritorio/3/SI_II/SI2/P2$ asadmin --host 10.8.6.2 --user admin --passwordfile ./passwordfile get config s.config.server-config.thread-pools.thread-pool.http-thread-pool.max-queue-size
configs.config.server-config.thread-pools.thread-pool.http-thread-pool.max-queue-size=4096
Command get executed successfully.
```

2. Maximum Pool Size del Pool de conexiones a nuestra DB

Hemos ejecutado el comando, obteniendo un tamaño máximo de 32 conexiones:

```
^Calba@alba-Aspire-E5-574:~/Escritorio/3/SI_II/SI2/P2$ asadmin --host 10.8.6.2 --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.
```

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

Podemos observar que ocurren 6 errores (columna EC, error count).

```
alba@alba-Aspire-E5-574:~/Escritorio/3/SI_II/SI2/P2$ asadmin --host 10.8.6.2 --user admin --passwordfile ./passwordfile monitor --type httpListener
ec mt pt rc
6 6418 235.00 198
6 6418 235.00 198
```

Ejercicio 5:

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

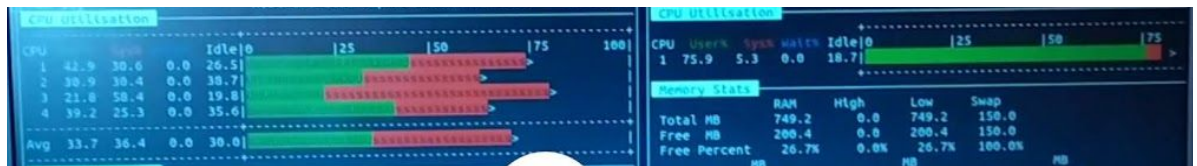
Elemento	Parámetro	Valor
VM Settings	Heap Max. (MB)	512
VM Settings	Heap Min. (MB)	512
HTTP Service	Max.Thread Count	5
HTTP Service	Queue size	4096
Web Container	Max.Sessions	-1
Visa Pool	Max.Pool Size	32

Ejercicio 6:

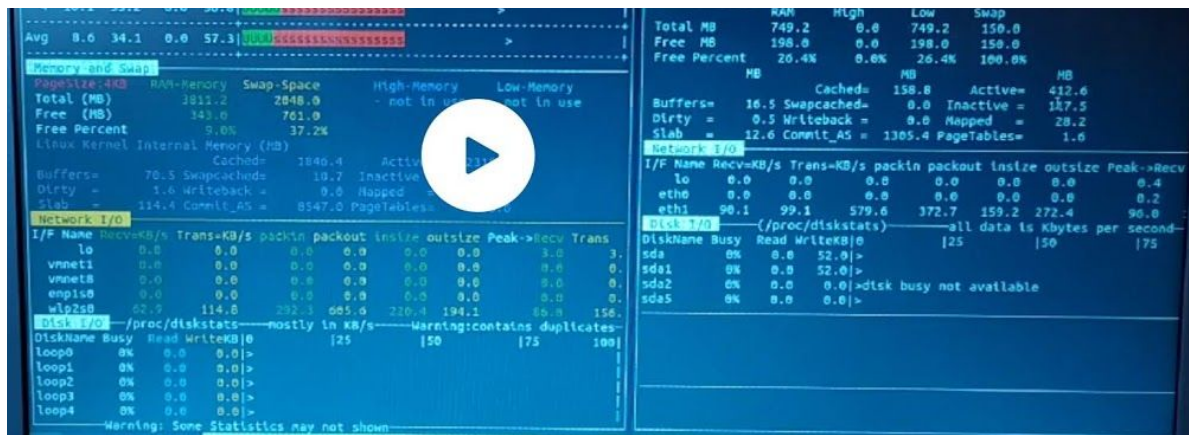
Tras habilitar la monitorización en el servidor, repita la ejecución del plan de pruebas anterior. Durante la prueba, vigile cada uno de los elementos de monitorización descritos hasta ahora. Responda a las siguientes cuestiones:

- 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 ...)

Observamos que el elemento más costoso es el consumo de CPU. Podemos observar que, en cuanto arranca la prueba, el uso de CPU tanto del usuario como del sistema en el PC1 se dispara (rojo y verde), y también se dispara en la máquina virtual 2 (pero en este caso el uso es del usuario, en verde):

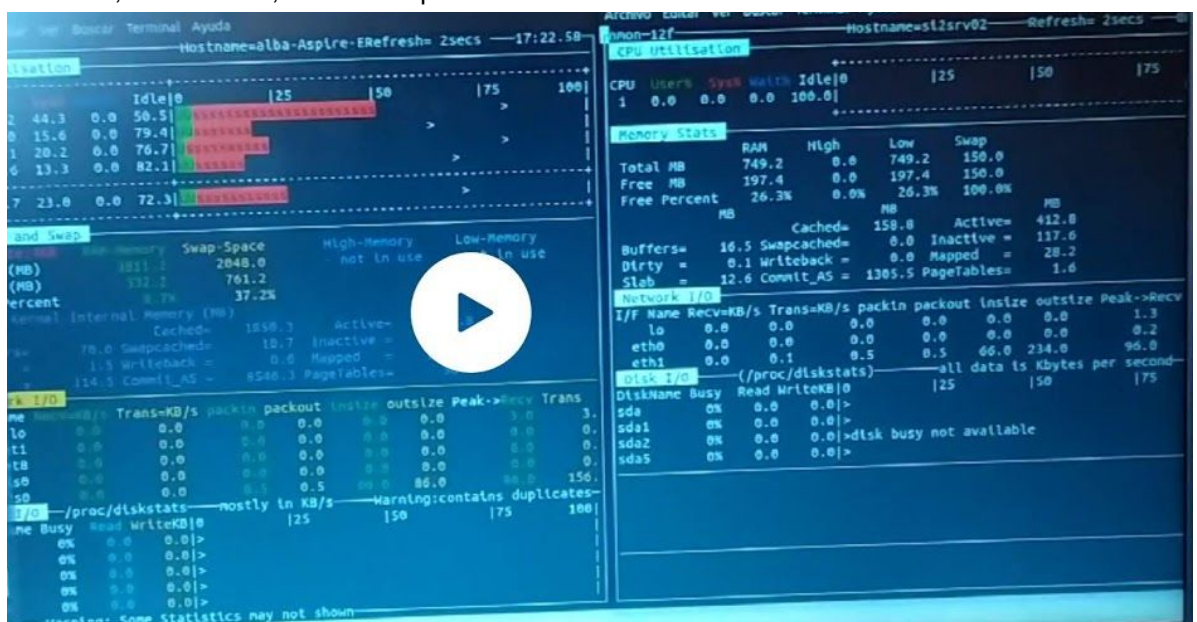


También vemos que la red se está usando a menudo (la interfaz wlp2s0), tanto en el PC1 como en la máquina virtual 2. Esto es lógico debido a que estamos realizando peticiones al servidor a través de esta interfaz. Vemos que el uso de red no es desmedido:



Sin embargo, en esta imagen observamos que la máquina virtual 2 está escribiendo en disco. Esto es un problema, ya que causará lentitud global ya que es un proceso muy costoso (pero tampoco es muy extraño debido a que la máquina virtual posee una memoria limitada, no es comparable a la memoria del PC1). Sin embargo, esto no es un comportamiento común, se ha observado que no ocurre a menudo.

En esta imagen vemos los valores una vez terminada la prueba. Como se puede comprobar, lo más llamativo es el descenso en el uso de CPU, nada comparado a lo que hemos visto antes. Es, en definitiva, el recurso que más se ha consumido.



Observamos que el script de monitorización muestra los siguientes resultados:

```

st2@st2srv02:~$ /bin/bash st2-monitor.sh localhost
#Muestra numJDBCCount numHTTPEndCount numHTTPOutCount
0 0 1 0
1 0 1 0
2 0 1 0
Me1oria P2 0 0 0 0
3 Editor Ver 0 0 0 0
4 0 1 0
5 0 1 0
6 0 1 0
7 1 1 0
8 0 1 0
9 0 1 0
10 1 1 0
11 1 1 0
12 0 1 0
13 0 1 0
14 1 1 0
15 0 1 0
16 0 1 0
17 0 1 0
18 1 1 0
19 0 1 0
20 0 1 0
21 0 0 0
22 0 0 0
23 0 0 0
24 0 0 0
25 0 0 0
26 0 0 0
27 0 0 0
28 0 0 0
29 0 0 0
30 0 0 0
31 0 0 0
32 0 0 0
33 0 0 0
34 0 0 0
35 0 0 0
#Le parece que situa
#Es realista
#Teniendo en cuenta
#despliegue de res
#No se restringe
34 0 0 0
35 0 0 0
AC
TOT.MUESTRAS MEDIA:
36 0.138889 0.583333 0
st2@st2srv02:~$

```

Observamos que en cada muestra tomada solo hay una única petición. Esto tiene sentido, ya que estamos realizando las 1000 pruebas usando un único hilo (como si un usuario realizara 1000 peticiones), por tanto no hay concurrencia.

- **¿Le parece una situación realista la simulada en este ejercicio? ¿Por qué?**
No es realista, porque estamos simulado que un mismo usuario realiza una única petición 1000 veces, pero en el mundo real esto no es muy común, lo común sería que diferentes usuarios realicen varias peticiones de forma concurrente. Aparte de esto, las peticiones se realizan de seguido y sin pausas entre ellas (cuando lo normal sería que el usuario realizara pausas entre peticiones).
- **Teniendo en cuenta cuál ha sido el elemento más saturado, proponga otro esquema de despliegue que resuelva esa situación.**
Para mejorar el rendimiento deberíamos, como recomienda en la práctica pero se nos es imposible, desplegar las distintas máquinas en distintos ordenadores, así la CPU no se ve tan sobrecargada.

Ejercicio 8:

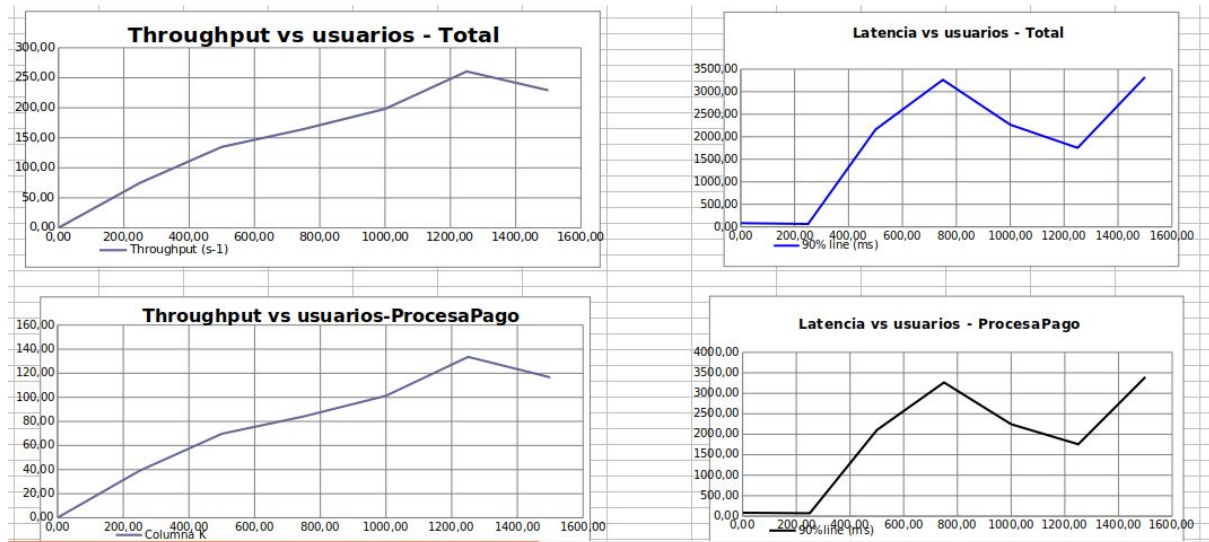
Obtener la curva de productividad, siguiendo los pasos que se detallan a continuación:

- 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 -> Clear All.
- 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” (Ver 8.2.2).
- 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, Aggregate Report.
- 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 (CPU average %). 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 CPU.
- 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.

A continuación se muestran las gráficas de productividad y latencia generadas a partir de las pruebas realizadas entre 1 y 1500 usuarios con incrementos de 250 en 250. Los datos obtenidos los podemos encontrar en el fichero SI2-P2-curvaProductividad.ods



Hemos tomado valores solamente hasta 1500 usuarios (donde ya vemos que se ha alcanzado el punto de saturación) debido a que al subir a 1750 usuarios el JMeter ha comenzado a mostrar excepciones debido a problemas en los sockets. Se muestra una imagen de las mismas. Creemos que esto se debe a que se ha alcanzado el máximo número de peticiones soportadas concurrentemente por el servidor, ya que hemos probado a lanzar 250 usuarios después de esto y no

ha habido ningún problema, quedando claro que el problema se encuentra en la cifra que estábamos intentando probar.

Nota: en el ejercicio 9 hemos vuelto a encontrar el problema y creemos que tenemos otra explicación.

Ayuda

00:01:35 2392 0/1750

Informe Agregado

Nombre:

Comentarios

Escribir todos los datos a Archivo

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

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Min	Máx	% Error	Rendimie...	Kb/sec	Sent KB/sec
/P1-base/...	3159	11276	3862	32780	47873	65841	1	68575	18,20%	34,4/sec	72,58	0,00
/P1-base/...	2322	15068	8207	39037	64388	65954	6	67715	26,74%	26,0/sec	32,85	0,00
	105	0	0	0	0	0	0	0	100,00%	∞/sec	0,00	0,00
Total	5586	12640	5402	33176	64076	65945	0	68575	23,29%	,0/hour	0,00	0,00

☐ ¿Incluir el nombre del grupo en la etiqueta? ☒ Guardar la cabecera de la tabla

```

1 | 2020-04-08 17:05:57,108 INFO o.a.j.t.JMeterThread: Thread finished: Consultas 1-1470
2 | 2020-04-08 17:05:57,109 ERROR o.a.j.p.h.s.HTTPJavaImpl: readResponse: java.net.SocketException: Socket closed
3 | 2020-04-08 17:05:57,109 ERROR o.a.j.p.h.s.HTTPJavaImpl: Cause: java.net.SocketException: Socket closed
4 | 2020-04-08 17:05:57,109 ERROR o.a.j.p.h.s.HTTPJavaImpl: readResponse: java.net.SocketException: Socket closed
5 | 2020-04-08 17:05:57,109 ERROR o.a.j.p.h.s.HTTPJavaImpl: Cause: java.net.SocketException: Socket closed
6 | 2020-04-08 17:05:57,109 INFO o.a.j.t.JMeterThread: Thread finished: Consultas 1-842
7 | 2020-04-08 17:05:57,109 ERROR o.a.j.p.h.s.HTTPJavaImpl: readResponse: java.net.SocketException: Socket closed

```

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.

Los datos de respaldo se encuentran en la carpeta ejercicio8 donde podemos encontrar capturas de la ejecución del script si2-monitor.sh para todos valores y del comando vmstat que se nos manda para la máquina virtual 2. Además también se encuentran las tablas creadas por el jmeter para cada uno de los valores.

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.

El punto de saturación se alcanza con 1250 usuarios, obteniendo un throughput de 260'52, siendo este además el throughput máximo que se obtiene en la zona de saturación.

Nota: se toman los valores de la gráfica "Throughput vs usuarios - Total"

- **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.**

Habíamos pensado dos alternativas:

- **Max. thread count:** pensamos que al poder soportar mayor concurrencia, habrá menos peticiones en la cola de espera y así se podrán atender a más peticiones a la vez, mejorando el rendimiento. Sin embargo, al realizar las pruebas hemos comprobado que esta no es la mejor alternativa.
- **Queue Size:** al haber una cola más grande, un mayor número de usuarios puede esperar en ella para ser atendidos, mejorando el rendimiento al poder atender a más usuarios (aunque no se les atiende simultáneamente, pero se les permite esperar su turno).
- **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.**

- **Max. thread count:** hemos probado esta alternativa y el rendimiento ha empeorado (se adjunta el fichero generado por JMeter *aggregate1250mejoradoHilos.csv*, donde se aprecia un rendimiento peor que antes para 1250 usuarios), y tras analizar la situación no es de extrañar: por mucho que aumentemos la cantidad de hilos concurrentes que puede haber, si nuestro ordenador no da para más el problema será el mismo. Incluso podría empeorar, ya que el ordenador se ve obligado a soportar un mayor número de hilos concurrentes.
- **Queue Size:** esta parece ser mejor alternativa, sin embargo cuando la hemos ido a probar han vuelto a ocurrir las excepciones descritas en el ejercicio 8. Hemos probado a reducir drásticamente la cantidad de usuarios con la que lo probábamos, y el problema ha continuado ocurriendo. Finalmente, hemos dejado el tamaño de la cola como estaba originalmente y hemos realizado otras pruebas, pero seguía ocurriendo el error. No hemos sido capaces de solucionarlo y no entendemos el motivo de su aparición.

A la vista de los valores de CPU observados durante las pruebas, podría ser que la máquina se esté saturando debido a que el consumo de CPU es muy elevado, sin embargo hasta 1500 usuarios en el ejercicio 8 no tuvimos ningún problema, y ahora en este ejercicio dejando todo como estaba y con 250 usuarios hay problemas, así que no estamos seguros de que sea por eso. Sea lo que sea, no hemos sido capaces de ejecutarlo incluso tras reiniciar nuestro PC.