
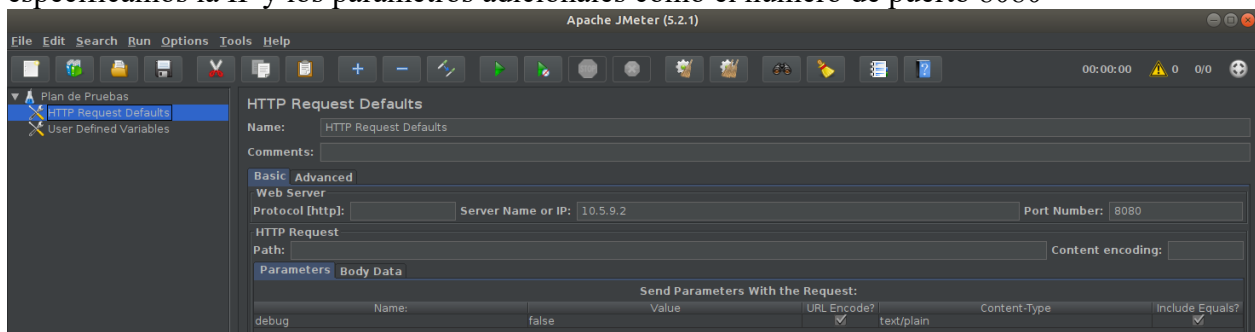


| | | | | | |
|---|-------------|---|---|--------------|------------|
|  | | Escuela Politécnica Superior Ingeniería Informática Prácticas de Sistemas Informáticos 2 | | | |
| Grupo | 2312 | Práctica | 2 | Fecha | 16/04/2021 |
| Alumno/a | | San Felipe Martín, Adrián | | | |
| Alumno/a | | Durán Díaz, Luis Miguel | | | |

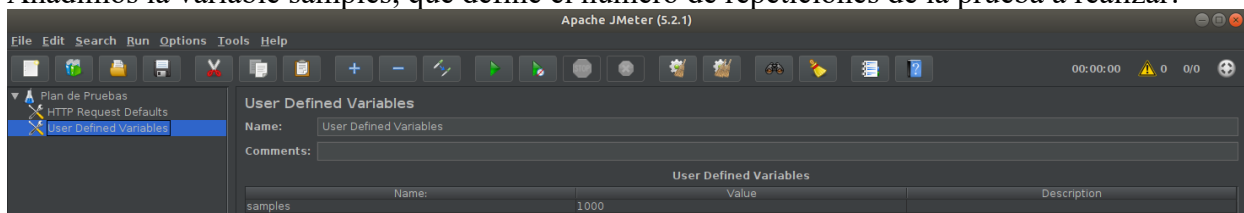
Ejercicio 1: Siguiendo todos los pasos anteriores, defina el plan completo de pruebas para realizar las tres ejecuciones secuenciales sobre los tres proyectos definidos hasta ahora (P1-base, P1-ws, P1-ejb). Adjunte el fichero generado P2.jmx al entregable de la práctica.

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.

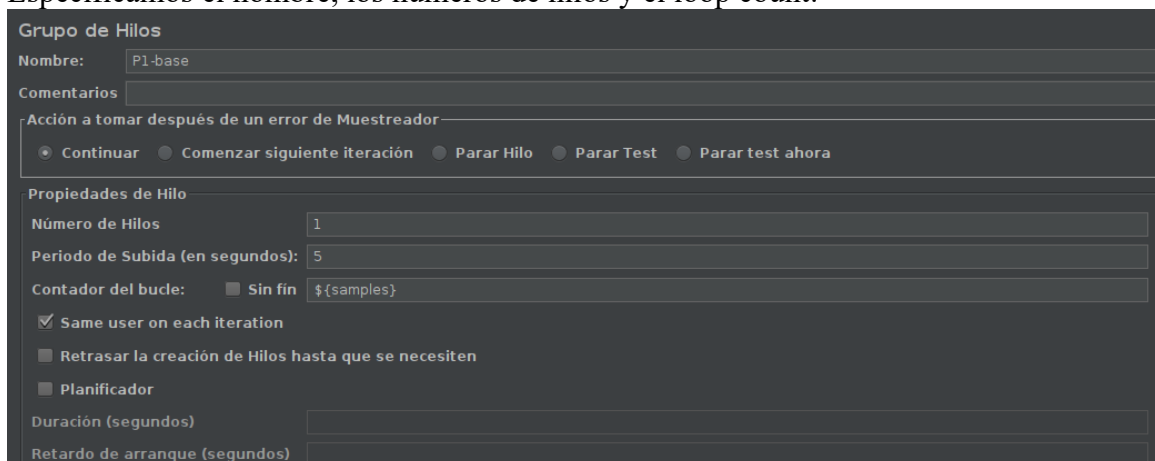
Añadimos el elemento de configuración HTTP por defecto, y en los valores por defecto especificamos la IP y los parámetros adicionales como el número de puerto 8080



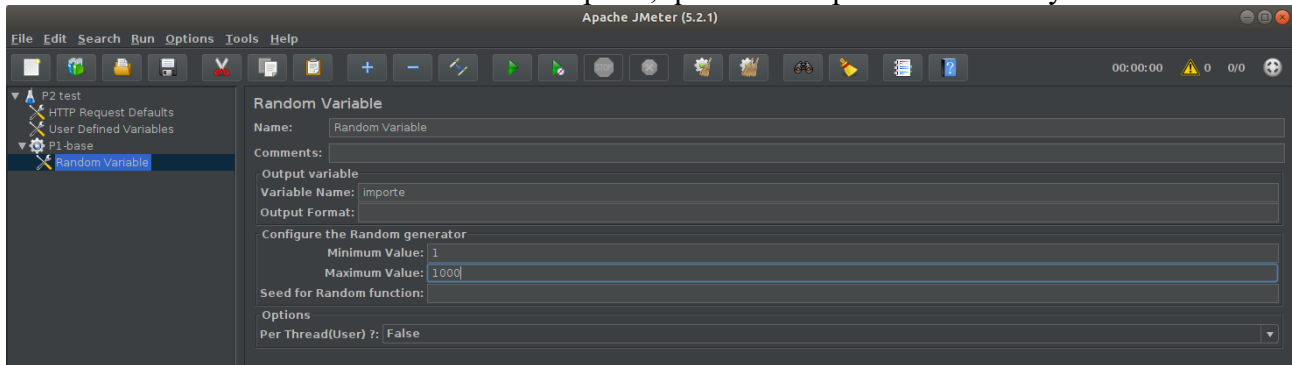
Añadimos la variable samples, que define el numero de repeticiones de la prueba a realizar:



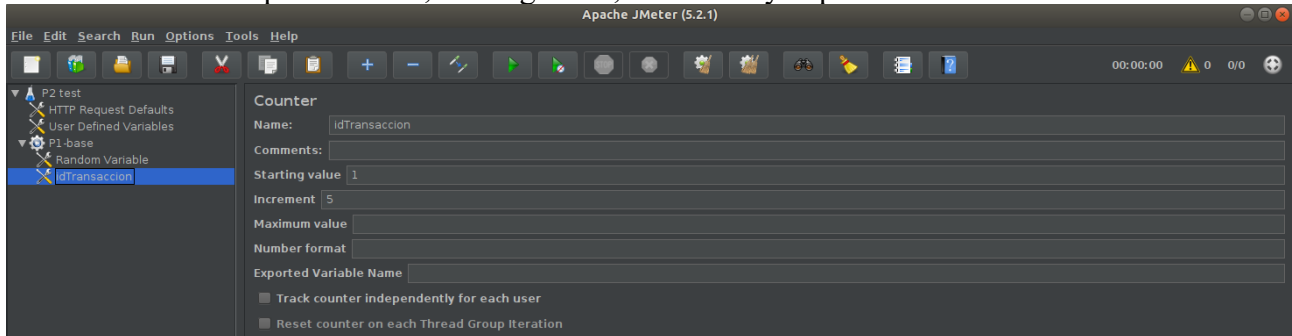
Especificamos el nombre, los números de hilos y el loop count:



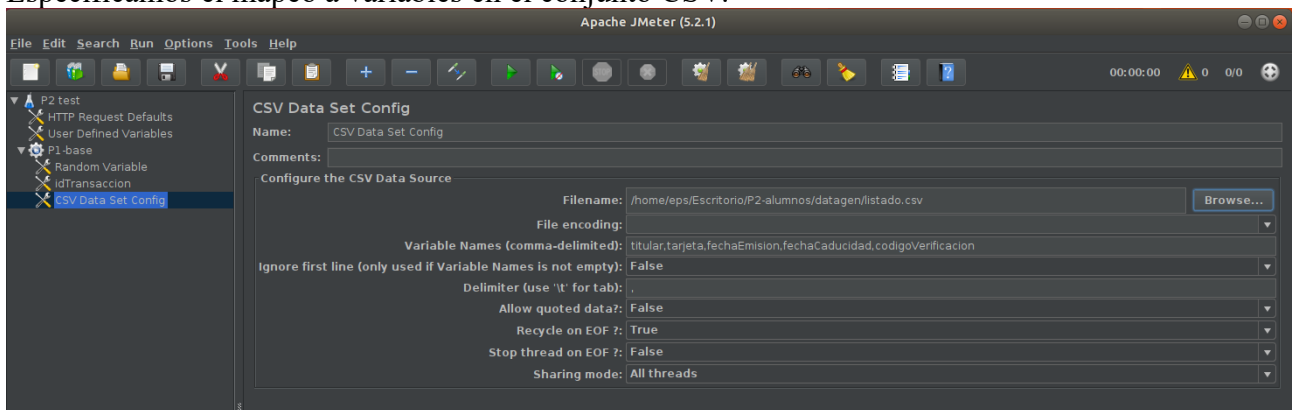
Añadimos la variable aleatoria de nombre importe, que está comprendida entre 1 y 1000:



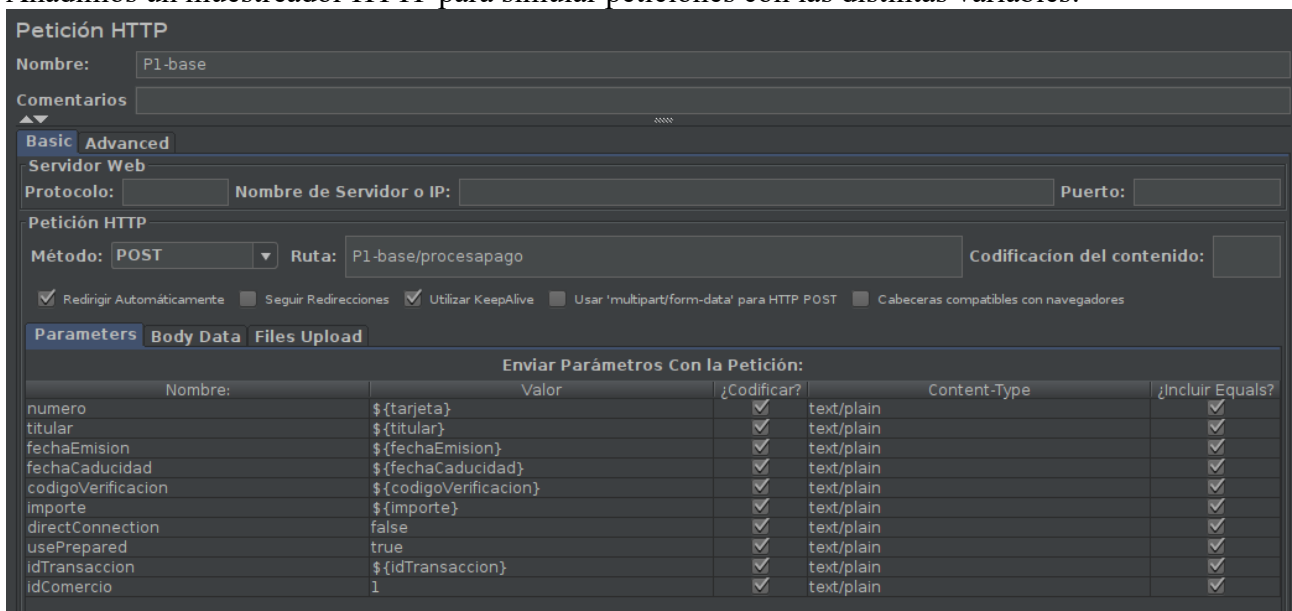
Rellenamos los campos de name, starting value, increment y exported variable name:



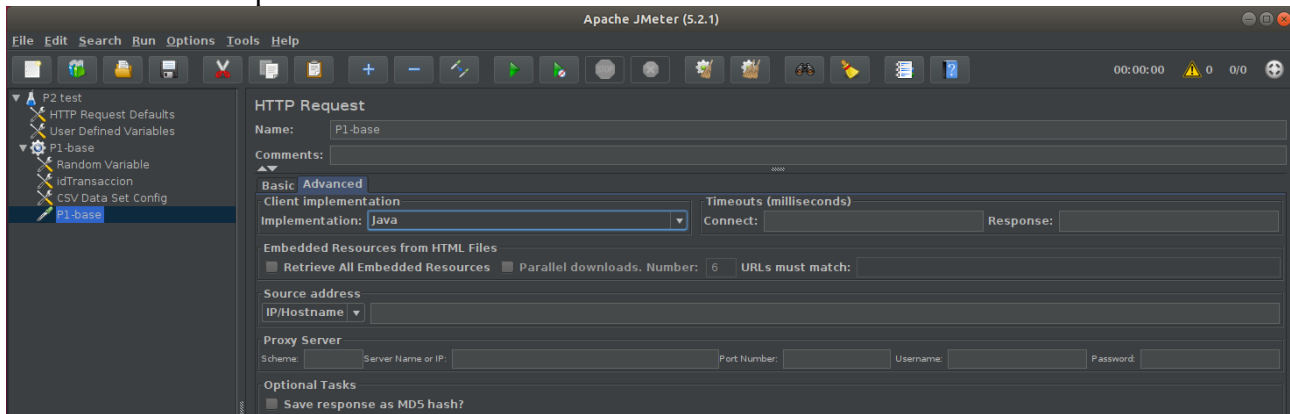
Especificamos el mapeo a variables en el conjunto CSV:



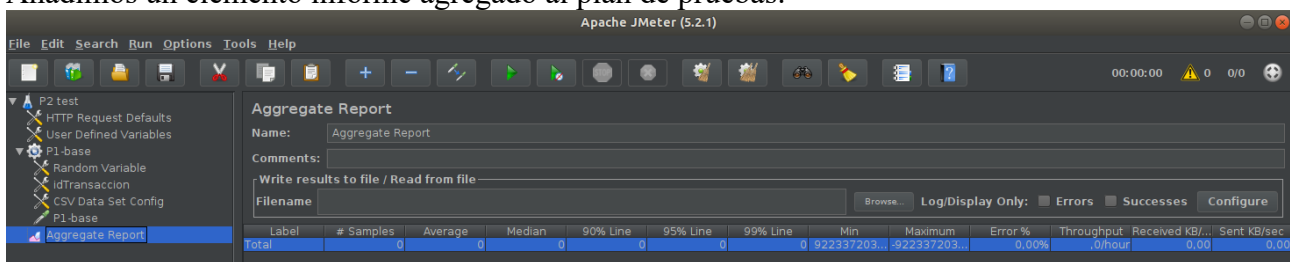
Añadimos un muestreador HTTP para simular peticiones con las distintas variables:



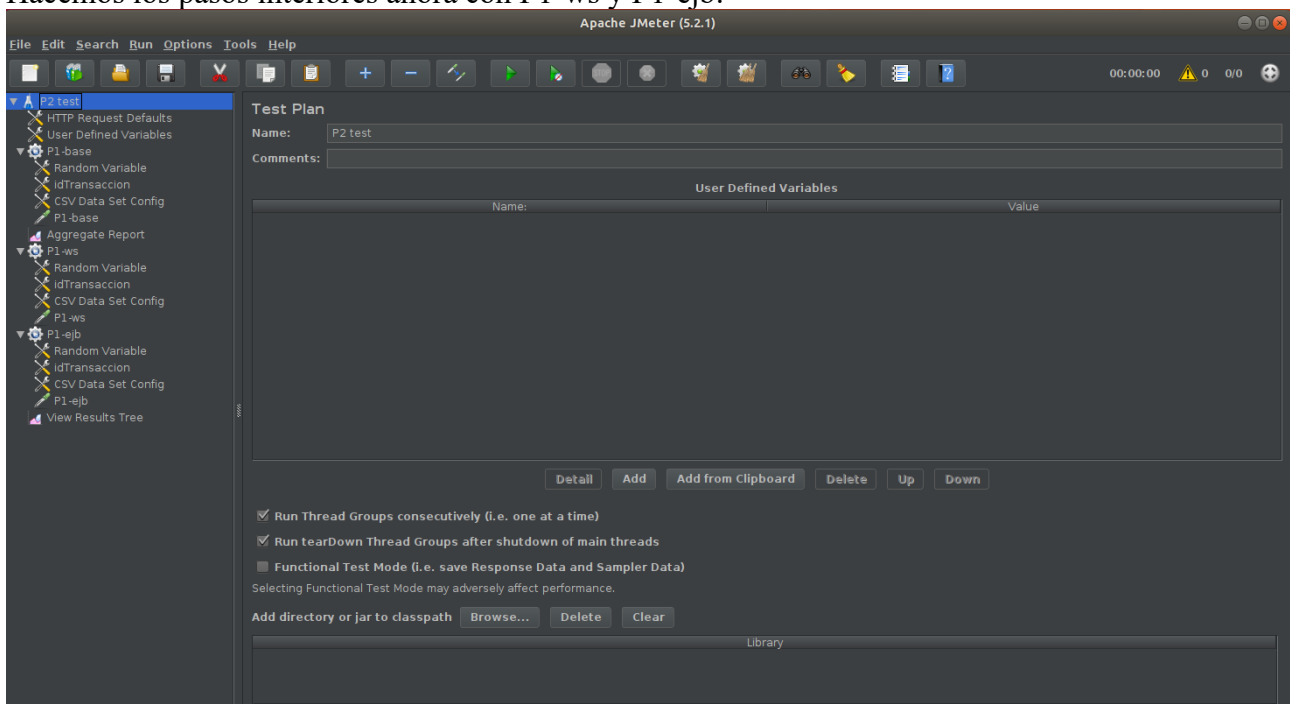
Establecemos la implementación del HTTP a Java:



Añadimos un elemento informe agregado al plan de pruebas:



Hacemos los pasos interiores ahora con P1-ws y P1-ejb:



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
- 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-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)
 - 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.

- Direcciones IP:
 - PC Host: 192.168.0.34 / 10.10.0.34
 - Máquina Virtual 1: 10.5.9.1
 - Máquina Virtual 2: 10.5.9.2
- Detener el servidor de Glassfish.
- Iniciar los servidores de Glassfish.
- Replegar las aplicaciones anteriores.
- Modificar los nombres de los ficheros build.properties:
 - Modificar nombre de P1 a P1-base (en P1-base).
 - En P1-ws no hace falta modificar el nombre (P1-ws).
 - En P1-ejb-cliente-remoto no hace falta modificar el nombre (P1-ejb-cliente-remoto).
 - En P1-ejb-servidor-remoto no hace falta modificar el nombre (P1-ejb).
- Modificar el fichero glassfish-web.xml:
 - Cambiamos la ip que aparecía anteriormente (terminada en 2), por la que se va a usar ahora (la ip 1):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE glassfish-web-app PUBLIC "-//GlassFish.org//DTD GlassFish Application Server 3.1 Servlet 3.0//EN" "http://glassfish.org/dtds/glassfish-web-app_3_0-1.dtd">
<glassfish-web-app>
  <ejb-ref>
    <ejb-ref-name>VisaDAOBean</ejb-ref-name>
    <jndi-name>corbaname:iiop:10.5.9.1:3700#java:global/P1-ejb/P1-ejb/VisaDAOBean!ssii2.visa.VisaDAORemote</jndi-name>
  </ejb-ref>
</glassfish-web-app>
```

- Cambiar el esquema de los build.properties (ip's) tal y como se nos indica en el enunciado.

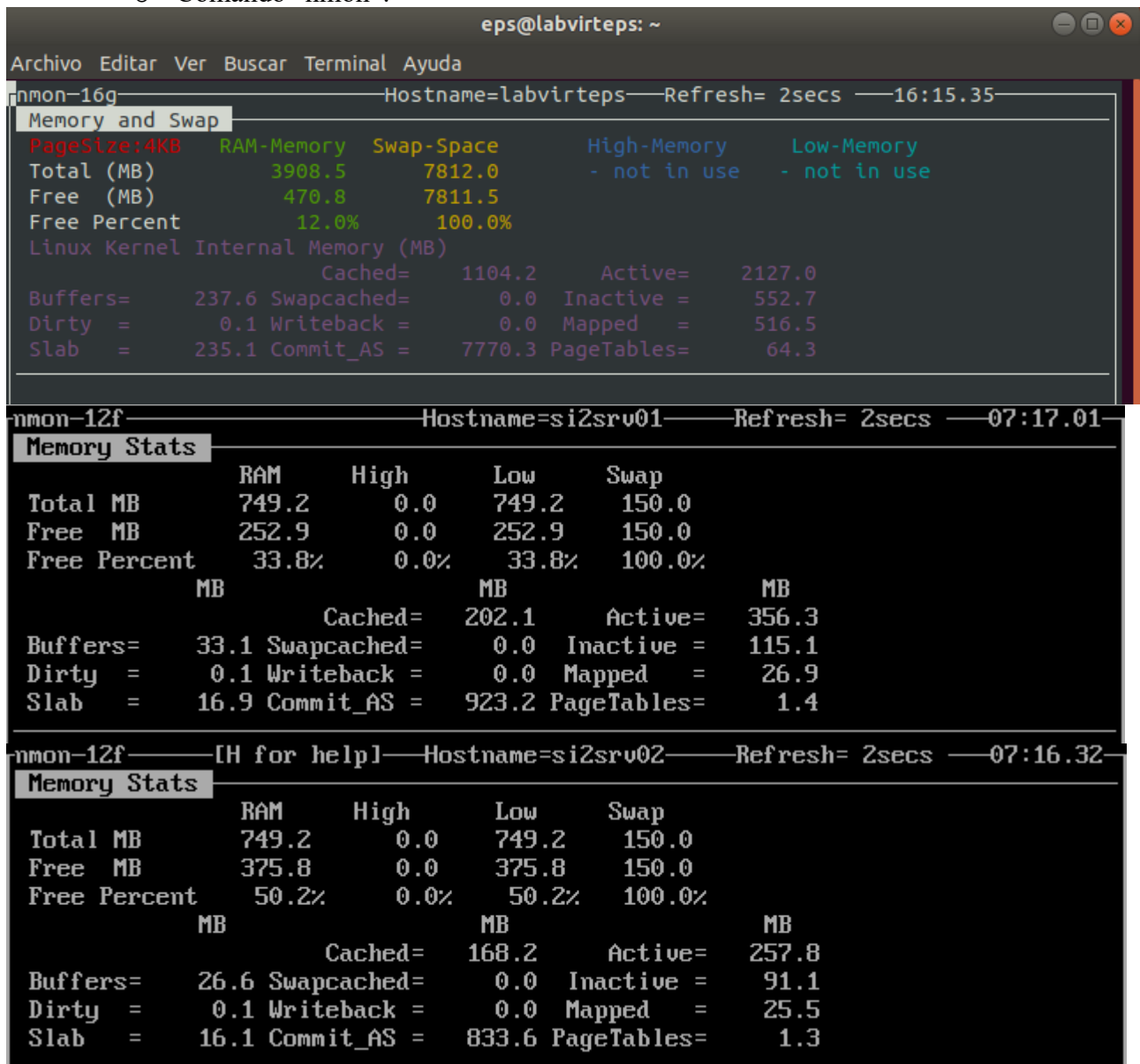
- Hacer “free” y “nmon” en la máquina “real” y las dos máquinas virtuales:
 - Comando “free”:

```
eps@labvirteps:~$ free
              total        usado        libre  compartido búfer/caché  disponible
Memoria:    4002268    2010740    486876      119852    1504652    1600836
Swap:       7999484        524    7998960

si2@si2srv01:~$ free
              total        used        free      shared    buffers    cached
Mem:        767168    506632    260536         0     33708    206836
-/+ buffers/cache:    266088    501080
Swap:       153592         0    153592

si2@si2srv02:~$ free
              total        used        free      shared    buffers    cached
Mem:        767168    379900    387268         0     27068    172092
-/+ buffers/cache:    180740    586428
Swap:       153592         0    153592
```

- Comando “nmon”:



The following screenshots show the output of the nmon command-line utility, which displays memory and swap statistics in a table format. The first screenshot shows the output for the host 'labvirteps', the second for 'si2srv01', and the third for 'si2srv02'. Each screenshot includes a title bar with the host name and a menu bar with options like 'Archivo', 'Editar', 'Ver', 'Buscar', 'Terminal', and 'Ayuda'.

Host: labvirteps

| Memory and Swap | |
|-----------------------------------|--|
| Page Size: 4KB | RAM-Memory Swap-Space High-Memory Low-Memory |
| Total (MB) | 3908.5 7812.0 - not in use - not in use |
| Free (MB) | 470.8 7811.5 |
| Free Percent | 12.0% 100.0% |
| Linux Kernel Internal Memory (MB) | |
| Cached= | 1104.2 Active= 2127.0 |
| Buffers= | 237.6 Swapcached= 0.0 Inactive = 552.7 |
| Dirty = | 0.1 Writeback = 0.0 Mapped = 516.5 |
| Slab = | 235.1 Commit_AS = 7770.3 PageTables= 64.3 |

Host: si2srv01

| Memory Stats | | | | |
|--------------|---------|-------------|---------|-------------|
| | RAM | High | Low | Swap |
| Total MB | 749.2 | 0.0 | 749.2 | 150.0 |
| Free MB | 252.9 | 0.0 | 252.9 | 150.0 |
| Free Percent | 33.8% | 0.0% | 33.8% | 100.0% |
| MB | | | | |
| | Cached= | 202.1 | Active= | 356.3 |
| Buffers= | 33.1 | Swapcached= | 0.0 | Inactive = |
| Dirty = | 0.1 | Writeback = | 0.0 | Mapped = |
| Slab = | 16.9 | Commit_AS = | 923.2 | PageTables= |

Host: si2srv02

| Memory Stats | | | | |
|--------------|---------|-------------|---------|-------------|
| | RAM | High | Low | Swap |
| Total MB | 749.2 | 0.0 | 749.2 | 150.0 |
| Free MB | 375.8 | 0.0 | 375.8 | 150.0 |
| Free Percent | 50.2% | 0.0% | 50.2% | 100.0% |
| MB | | | | |
| | Cached= | 168.2 | Active= | 257.8 |
| Buffers= | 26.6 | Swapcached= | 0.0 | Inactive = |
| Dirty = | 0.1 | Writeback = | 0.0 | Mapped = |
| Slab = | 16.1 | Commit_AS = | 833.6 | PageTables= |

- Realizar pagos de calentamiento:

Sistema de Pago con tarjeta x Sistema de Pago con tarjeta x

No es seguro | 10.5.9.2:8080/P1-base/testbd.jsp

Aplicaciones UAM - Escuela... MOODLE GRA...

Pago con tarjeta

Proceso de un pago

Id Transacción:

Id Comercio:

Importe:

Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

Modo debug: ☐ True ☐ False

Direct Connection: ☐ True ☐ False

Use Prepared: ☐ True ☐ False

Sistema de Pago con tarjeta x Sistema de Pago con tarjeta x

No es seguro | 10.5.9.2:8080/P1-ws-cliente/testbd.jsp

Aplicaciones UAM - Escuela... MOODLE GRA... MO

Pago con tarjeta

Proceso de un pago

Id Transacción:

Id Comercio:

Importe:

Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

Modo debug: ☒ True ☐ False

Direct Connection: ☒ True ☐ False

Use Prepared: ☒ True ☐ False

Sistema de Pago con tarjeta x Sistema de Pago con tarjeta x Sist

No es seguro | 10.5.9.2:8080/P1-ejb-cliente-remoto/testbd.jsp

Aplicaciones UAM - Escuela... MOODLE GRA... MOODLE F

Pago con tarjeta

Proceso de un pago

Id Transacción:

Id Comercio:

Importe:

Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

Modo debug: ☐ True ☐ False

Direct Connection: ☐ True ☐ False

Use Prepared: ☐ True ☐ False

Sistema de Pago con tarjeta x GlassFish Console - Com

No es seguro | 10.5.9.2:8080/P1-ejb-cliente-rem

Aplicaciones UAM - Escuela... MOODLE GRA...

Pago con tarjeta

Lista de pagos del comercio 1

| idTransaccion | Importe | codRespuesta | idAutorizacion |
|---------------|---------|--------------|----------------|
| 1 | 100.0 | 000 | 1 |
| 2 | 200.0 | 000 | 2 |
| 3 | 300.0 | 000 | 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.

Una vez que los resultados han sido satisfactorios:

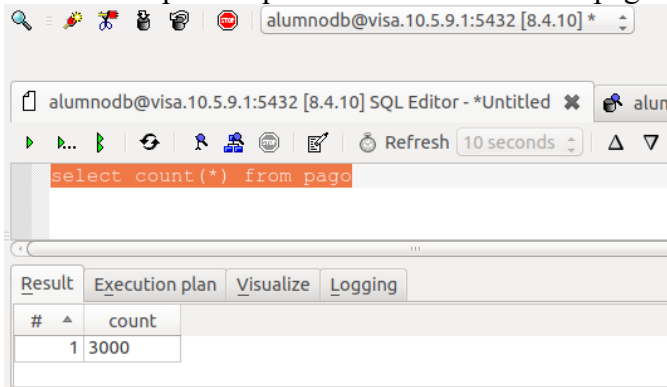
- Anote los resultados del informe agregado en la memoria de la práctica.
- Salve el fichero server.log que se encuentra en la ruta glassfish/domains/domain1/logs de Glassfish y adjúntelo con la práctica.
- Añada a la memoria de prácticas la siguiente información: ¿Cuál de los resultados le parece el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

Incluir el directorio P2 en la entrega.

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.

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

- Comprobar que se han realizado 3000 pagos:



- Comprobar que no ha habido ningún error:

| Etiqueta | # Muestras | Media | Mediana | 90% Line | 95% Line | 99% Line | Min | Máx | % Error | Rendimiento | Kb/sec | Sent KB/sec |
|----------|------------|-------|---------|----------|----------|----------|-----|------|---------|-------------|--------|-------------|
| P1-base | 1000 | 8 | 5 | 7 | 9 | 11 | 3 | 3301 | 0,00% | 114,1/sec | 87,06 | 0,00 |
| P1-ws | 1000 | 46 | 40 | 52 | 58 | 72 | 34 | 4560 | 0,00% | 21,2/sec | 16,27 | 0,00 |
| P1-ejb | 1000 | 15 | 10 | 15 | 17 | 19 | 7 | 5006 | 0,00% | 62,3/sec | 48,35 | 0,00 |
| Total | 3000 | 23 | 10 | 44 | 49 | 64 | 3 | 5006 | 0,00% | 41,6/sec | 32,02 | 0,00 |

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

El mejor resultado que nos parece es el de P1-base, ya que su Media es de 8, que en comparación con P1-ejb es casi la mitad y 6 veces menos que P1-ws, además de que podemos observar también que el rendimiento de P1-base es el mejor de los tres.

Pensamos que se debe a que en P1-base el servidor y el cliente se encuentran en la misma máquina virtual, por lo que el acceso a los recursos y datos es más rápido.

Para decidir este resultado hemos elegido la columna del rendimiento.

- Repetir la prueba de P1-ejb:

| Etiqueta | # Muestras | Media | Mediana | 90% Line | 95% Line | 99% Line | Min | Máx | % Error | Rendimiento | Kb/sec | Sent KB/sec |
|----------|------------|-------|---------|----------|----------|----------|-----|-----|---------|-------------|--------|-------------|
| P1-ejb | 1000 | 10 | 10 | 12 | 13 | 15 | 8 | 27 | 0,00% | 93,3/sec | 72,47 | 0,00 |
| Total | 1000 | 10 | 10 | 12 | 13 | 15 | 8 | 27 | 0,00% | 93,3/sec | 72,47 | 0,00 |

En este caso podemos observar que el rendimiento ha mejorado bastante, por tanto la media también. Esto es por la misma razón que antes hemos comentado respecto a P1-base, el cliente, el servidor y la base de datos se encuentran en la misma máquina virtual (en este caso, es la que tiene la IP 1)

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`³. 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`⁴ que debemos ejecutar en el PC host para averiguar los valores siguientes, mencionados en el Apéndice 1, del servidor PC1VM1:

1. Max Queue Size del Servicio HTTP
2. 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.

Añadimos `-Xms512m`, `-server` y borramos `-client`:

The screenshot shows the GlassFish administration console interface. On the left, a tree view lists various configuration categories, with 'JVM Settings' selected under the 'Configurations' section. The main panel, titled 'Options (32)', displays a list of JVM options. Each option has a checkbox in the 'Select' column and its corresponding value in the 'Value' column. The options include memory settings like `-Xms512m` and `-Xmx512m`, JVM flags like `-server` and `-XX:MaxPermSize=192m`, and various system properties starting with `-D` for paths and security configurations. The interface also shows user information (admin, domain1) and server version (10.5.9.2) at the top.

Desmarcamos el check de Reload y Auto Deploy:

Home About...

User: admin Role: domain1 Server: 10.5.9.2

GlassFish™ Server Open Source Edition

Total # of available updates : 1

Tree

- Common Tasks
- Domain
 - server (Admin Server)
 - Clusters
 - Standalone Instances
 - Nodes
 - Applications
 - Lifecycle Modules
 - Monitoring Data
 - Resources
 - Concurrent Resources
 - Connectors
 - JDBC
 - JMS Resources
 - JNDI
 - JavaMail Sessions
 - Resource Adapter Configs
 - Configurations
 - default-config
 - server-config
 - Admin Service
 - Connector Service
 - EJB Container
 - HTTP Service
 - JVM Settings
 - Java Message Service
 - Logger Settings

Applications Configuration

Enable reloading so that changes to deployed applications are detected and the modified classes reloaded. Also enable and configure :

Load Defaults

Reload: ☐ Enabled
Enables dynamic reloading of applications.

Reload Poll Interval: 2 Seconds
Frequency for checking reload requests.

Admin Session Timeout: 60 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: 2 Seconds
Frequency at which the autodeploy directory is checked for applications; interval does not affect amou

Auto Deploy Retry Timeout: 4 Seconds
Time to report failure after a file remains stable in size but cannot be opened.

Auto Deploy Directory: \${com.sun.aas.instanceRoot}/autodeploy
Directory to monitor for autodeploy applications.

XML Validation: Full
Type of deployment descriptor validation.

Verifier: ☐ Enabled
Performs detailed verification before deployment.

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

Ponemos en HIGH los servicios indicados:

Home About...

User: admin Role: domain1 Server: 10.5.9.2

GlassFish™ Server Open Source Edition

Total # of available updates : 1

Tree

- Common Tasks
- Domain
 - server (Admin Server)
 - Clusters
 - Standalone Instances
 - Nodes
 - Applications
 - Lifecycle Modules
 - Monitoring Data
 - Resources
 - Concurrent Resources
 - Connectors
 - JDBC
 - JMS Resources
 - JNDI
 - JavaMail Sessions
 - Resource Adapter Configs
 - Configurations
 - default-config
 - server-config
 - Admin Service
 - Connector Service
 - EJB Container
 - HTTP Service
 - JVM Settings
 - Java Message Service
 - Logger Settings
 - Monitoring
 - Network Config
 - ORB

Monitoring Service

Enable monitoring for a component or service by selecting either LOW or HIGH. Monitoring Service and Monitoring MBeans must both be enabled to use Administration Console monitoring features.

Configuration Name: server-config

Monitoring Service: ☒ Enabled
Enable monitoring for GlassFish Server

Monitoring MBeans: ☒ Enabled
Deploy all MBeans needed for monitoring

Component Level Settings (16)

| 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 |

1. Max Queue Size del Servicio HTTP

asadmin --host 10.5.9.2 --user admin --passwordfile ./passwordfile get -m server.http-service.connection-pool.max-pending-count

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

asadmin --host 10.5.9.2 --user admin --passwordfile ./passwordfile get -m domain.resources.jdbc-connection-pool.myjdbc_oracle-pool

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

Parámetros de configuración

| Elemento | Parámetro | Valor | Descripción de la prueba |
|---------------|------------------|-------|---|
| JVM Settings | Heap Máx. (MB) | 512 | Los valores de cada uno de los parámetros se han tomado siguiendo las instrucciones indicadas en el enunciado de la práctica. |
| 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:

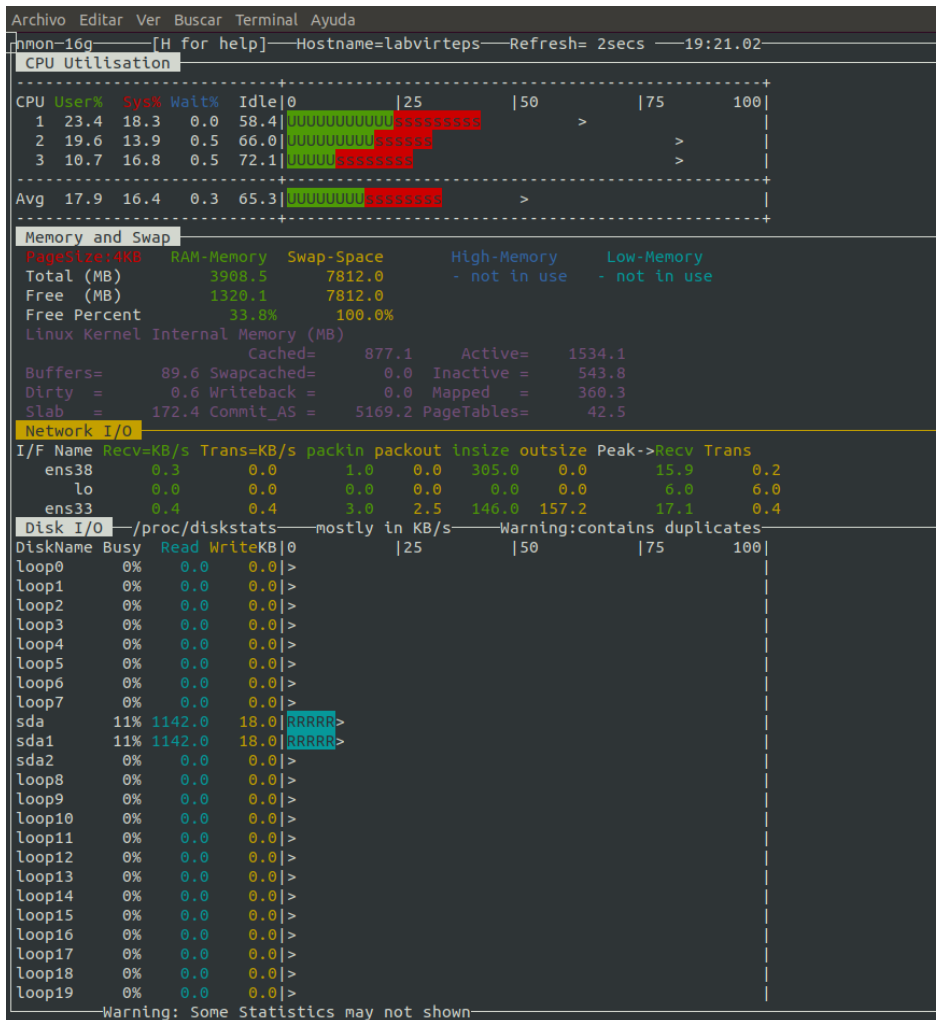
- A la vista de los resultados, ¿qué elemento de proceso le parece más costoso? ¿Red? ¿CPU? ¿Acceso a datos? En otras palabras, ¿cuál fue el elemento más utilizado durante la monitorización con *nmon* en un entorno virtual? (CPU, Memoria, disco ...)
- ¿Le parece una situación realista la simulada en este ejercicio? ¿Por qué?
- Teniendo en cuenta cuál ha sido el elemento más saturado, proponga otro esquema de despliegue que resuelva esa situación.
-

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*.

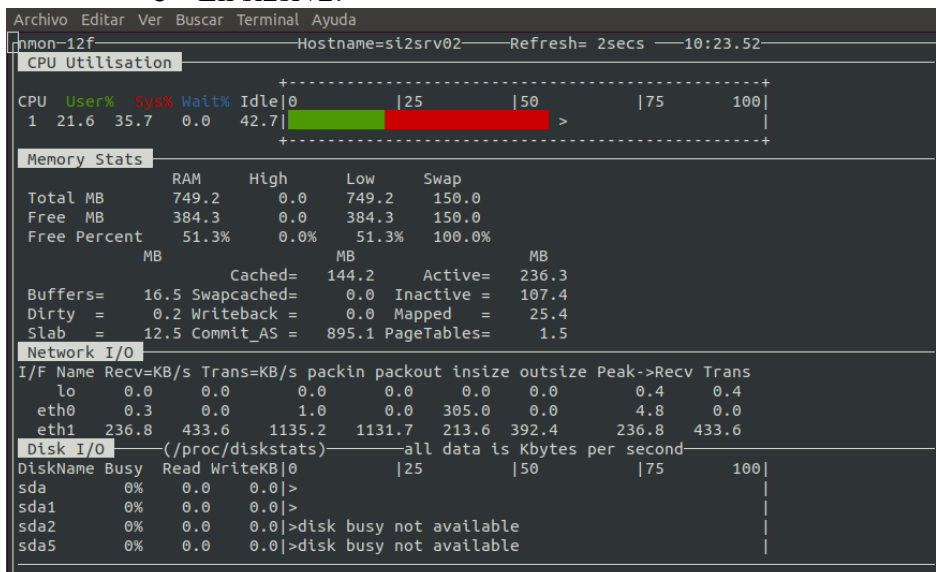
Al ejecutar el script desde el servidor 2, podemos observar como se están ejecutando las peticiones y consultas correctamente:

```
si2@si2srv02:~$ ./si2-monitor.sh localhost
#Muestra numJDBCCount numHTTPCount numHTTPQ
0 0 0 0
1 0 1 0
2 1 1 0
3 0 0 0
4 0 1 0
5 1 1 0
6 0 0 0
7 0 0 0
8 0 0 0
9 0 0 0
10 0 0 0
11 0 0 0
12 0 0 0
^C
TOT.MUESTRAS MEDIA:
13 0.153846 0.307692 0
```

- Nmon con opciones c, m, n, d:
 - En el PC host:



- En si2srv2:



- A la vista de los resultados, ¿Qué elemento le parece más costoso? ¿Red? ¿CPU? ¿Acceso a datos? En otras palabras, ¿Cuál fue el elemento más utilizado durante la monitorización?

A la vista de los resultados, el elemento que más carga de trabajo tiene que realizar es la CPU

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

No nos parece una situación realista, ya que en la prueba, solo hay un usuario ejecutándola, y lo más normal es que en un servicio siempre haya más de un usuario.

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

Para resolver el consumo alto de CPU, podríamos aumentar el número de núcleos disponibles o la velocidad de estos.

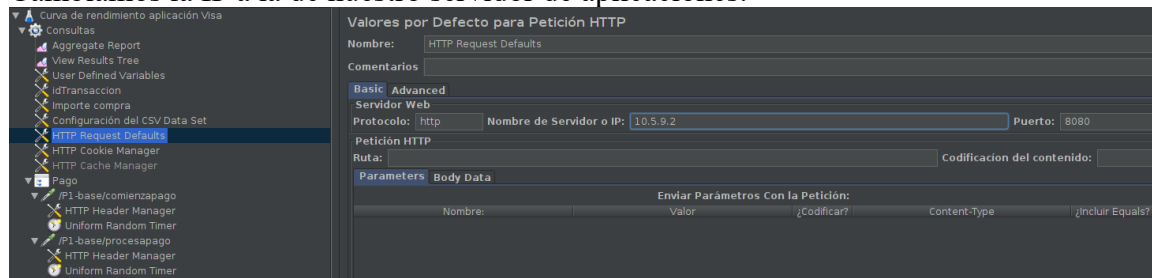
Ejercicio 7: Preparar el *script* de JMeter para su ejecución en el entorno de pruebas. Cambiar la dirección destino del servidor para que acceda al host en el que se encuentra el servidor de aplicaciones. Crear también el directorio *datagen* en el mismo directorio donde se encuentre el *script*, y 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.

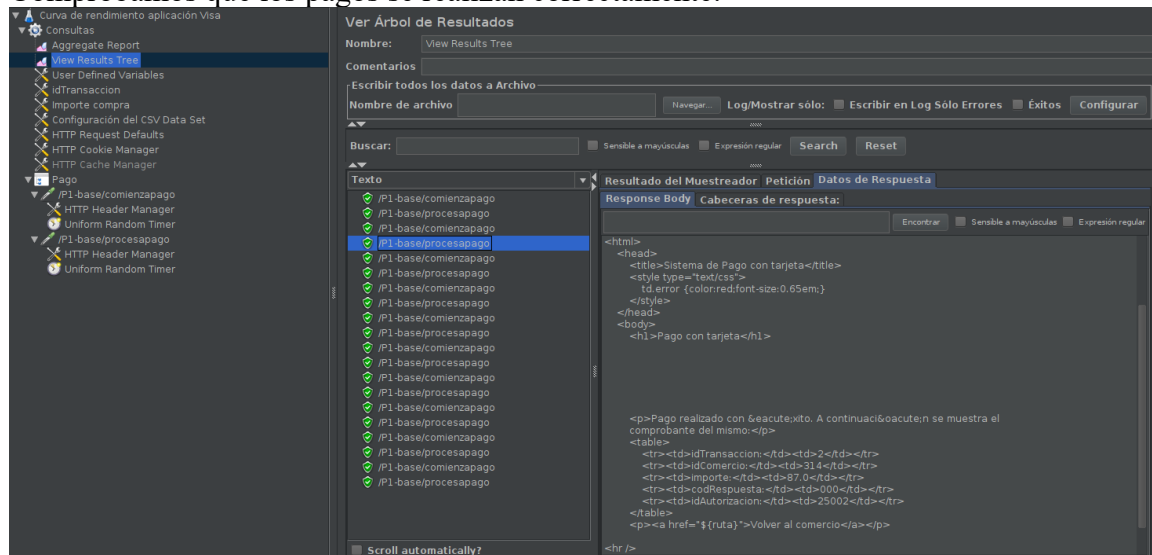
Cambiamos la IP a la de nuestro servidor de aplicaciones:



Cambiamos el *thinkTimeMin* y *Max*:



Comprobamos que los pagos se realizan correctamente:



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 al 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 ⁵.
- 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*.

Todas las pruebas de las monitorizaciones se encuentran en la carpeta "EJ8". A su vez, dentro de la carpeta "P2" se encuentra el archivo de la curva de productividad para su visualización.

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.
- 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.
- 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.

- **Pregunta nº1:**

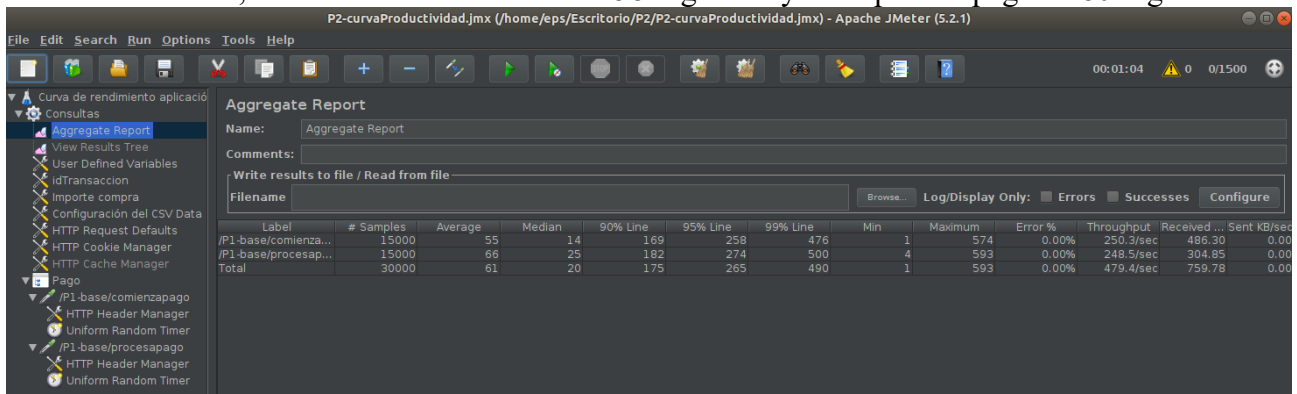
El punto de saturación se produce cuando pasamos de 1000 a 1500 usuarios. El *throughput* que se alcanza en ese punto es de 158 segundos y el que se obtiene en la zona de saturación es de 201 segundos.

- **Pregunta nº2:**

El parámetro que habría que cambiar para mejorar los resultados anteriores y así obtener un punto de saturación con mayor número de usuarios es la cantidad de procesadores, ya que la máquina virtual solo posee un procesador.

- **Pregunta nº3:**

Podemos observar que, después de aumentar el número de CPUs, el rendimiento ha mejorado muchísimo. Ahora el total son 479 segundos y de procesamiento son 248 segundos. En comparación, con solo una CPU, el rendimiento total era de 158 segundos y el de procesamiento de 80 segundos.



También podemos observar que el uso medio de CPU ha bajado casi la mitad:

```
si2@si2srv02:~$ vmstat -n 1 | (trap 'INT; awk '{print; if(NR>2) cpu+=$13+$14;}END{print "MEDIA"; print NR: ", NR, "CPU:", cpu/(NR-2);}');)
```

| Cprocs | | memory | | | | swap | | io | | system | | cpu | | | |
|--------|---|--------|---------|-------|--------|------|----|-----|----|--------|------|-----|----|----|----|
| r | b | swpd | free | buff | cache | si | so | bi | bo | in | cs | us | sy | id | wa |
| 5 | 0 | 0 | 2488256 | 17356 | 192304 | 0 | 0 | 168 | 16 | 344 | 418 | 6 | 6 | 87 | 0 |
| 4 | 0 | 0 | 2479104 | 17356 | 192464 | 0 | 0 | 0 | 0 | 4865 | 7067 | 45 | 28 | 26 | 0 |
| 3 | 0 | 0 | 2479404 | 17356 | 192632 | 0 | 0 | 0 | 0 | 5295 | 7838 | 32 | 30 | 38 | 0 |
| 0 | 0 | 0 | 2479160 | 17364 | 192772 | 0 | 0 | 0 | 32 | 4886 | 6062 | 33 | 19 | 48 | 0 |
| 1 | 0 | 0 | 2478504 | 17364 | 192920 | 0 | 0 | 0 | 0 | 5063 | 6919 | 39 | 20 | 42 | 0 |
| 3 | 0 | 0 | 2478360 | 17364 | 193080 | 0 | 0 | 0 | 0 | 5131 | 7043 | 37 | 24 | 39 | 0 |
| 1 | 0 | 0 | 2477896 | 17364 | 193224 | 0 | 0 | 0 | 0 | 4977 | 6763 | 29 | 27 | 44 | 0 |

MEDIA
NR: 9 CPU: 53.5714

Los valores del monitor también han mejorado considerablemente:

| | | | | |
|--------------|---------|--------|---------|----|
| | 21 | 5 | 5 | 2 |
| | 22 | 5 | 5 | 8 |
| | 23 | 2 | 5 | 68 |
| | 24 | 4 | 5 | 24 |
| | 25 | 4 | 5 | 16 |
| | 26 | 3 | 5 | 44 |
| | 27 | 0 | 5 | 4 |
| | 28 | 4 | 5 | 53 |
| | 29 | 3 | 5 | 34 |
| | 30 | 4 | 3 | 8 |
| | 31 | 3 | 5 | 41 |
| | 32 | 2 | 5 | 46 |
| | 33 | 1 | 5 | 22 |
| | 34 | 4 | 5 | 45 |
| | 35 | 2 | 5 | 4 |
| | 36 | 4 | 5 | 36 |
| | 37 | 4 | 5 | 35 |
| | 38 | 4 | 5 | 8 |
| | 39 | 4 | 5 | 21 |
| | 40 | 4 | 5 | 39 |
| | 41 | 0 | 5 | 12 |
| | 42 | 4 | 4 | 0 |
| | 43 | 0 | 0 | 15 |
| | 44 | 1 | 5 | 0 |
| | 45 | 0 | 0 | 0 |
| | 46 | 0 | 0 | 1 |
| | 47 | 2 | 0 | 0 |
| ^C | | | | |
| TOT.MUESTRAS | | MEDIA: | | |
| 48 | 2.85417 | 4.5 | 54.9167 | |