		<b>Escuela Politécnica Superior</b> <b>Ingeniería Informática</b> <b>Prácticas de Sistemas Informáticos 2</b>			
<b>Grupo</b>	<b>2311</b>	<b>Práctica</b>	<b>2</b>	<b>Fecha</b>	<b>04/04/2022</b>
<b>Alumno/a</b>		Fraile, Iglesias, Javier			
<b>Alumno/a</b>		Fernández, París, Iván			

## Práctica 2: Rendimiento

### Ejercicio número 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**

Además de crear el plan de pruebas, realizamos un cambio al nombre de la aplicación que se despliega con P1-base, cambiando la línea "nombre=P1" del fichero build.properties por "nombre=P1-base".

No se incluirán imágenes que reflejen que hemos llevado a cabo todos los pasos para definir un plan de pruebas puesto que el plan definido se entrega con el nombre "P2.jmx" y se puede abrir dicho plan para comprobar que se han seguido todas las pautas indicadas en el enunciado de la práctica P2.

### Ejercicio número 2:

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

En el test de pruebas realizamos un cambio

- **Anote en la memoria de prácticas las direcciones IP asignadas a las máquinas virtuales y al PC**  
Como se comenta en el apéndice 7.1 al ordenador PC1VM se le tiene que asignar la IP 10.X.Y.1 y al ordenador PC2VM se le tiene que asignar la IP 10.X.Y.2, en nuestro caso dichas IPs son 10.4.6.1 y 10.4.6.2 respectivamente.
- **Detenga el servidor de GlassFish del PC host**  
Con el comando "asadmin stop-domain domain1" detenemos el servidor de GlassFish.
- **Inicie los servidores GlassFish en las máquinas virtuales**  
Con el comando "asadmin start-domain domain1" iniciamos el servidor de GlassFish.
- **Repliegue todas las aplicaciones o pruebas anteriores (P1-base, P1-ws, etc), para limpiar posibles versiones incorrectas.**  
En la carpeta "P2-alumnos" se encuentran las versiones realizadas en las prácticas anteriores, y entrando una a una realizamos un repliegue de todas las aplicaciones con el comando "ant replegar".
- **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.**  
Como se comentó en el ejercicio anterior, de acuerdo con lo que indicaba el enunciado, cada versión tenía que tener un nombre específico y tuvimos que cambiar el de P1-base a "P1-base" modificando la línea "nombre=P1" del fichero build.properties.
- **Revise y modifique si es necesario el fichero glassfish-web.xml, para indicar la IP del EJB remoto que usa P1-ejb-cliente.**  
En el fichero glassfish-web.xml se encontraba indicada la IP 10.4.6.2 por lo que la cambiamos a la dirección 10.4.6.1
- **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)**  
Accedemos al fichero build.properties y modificamos el destino del despliegue a 10.4.6.2 (línea en la que aparece "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)**  
Respecto a la versión P1-ws, está ya se encontraba con dicha IP en el valor de la variable "as.host.client" por lo que no tuvimos que realizar ningún cambio. Mientras que en el caso de la versión P1-ejb-cliente-remoto sí tuvimos que modificar el fichero build.properties para cambiar la dirección del "as.host" de 10.4.6.1 a 10.4.6.2
- **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)**  
En este caso observamos la misma situación que en el caso anterior, el fichero build.properties de la versión P1-ws no se tuvo que modificar, mientras que en el caso de la versión P1-ejb-servidor-remoto si, teniendo que cambiar la dirección de 10.4.6.2 a 10.4.6.1 y además teniendo que añadir esas dos variables puesto que en nuestro caso solo usabamos "as.host".
- **La base de datos en todos ellos será la de PC1VM con IP 10.X.Y.1 (db.host)**  
Revisando cada uno de los ficheros "postgresql.properties" de cada versión, no hubo que realizar ningún cambio respecto a la dirección del host de la base de datos.  
Aprovechando que en este fichero se encuentra la dirección del cliente tuvimos que realizar una serie de modificaciones de las direcciones del "db.client.host" en consecuencia de los cambios anteriores. Tuvimos que modificar el fichero de la versión P1-base para indicar que la dirección del cliente ("db.client.host") no era 10.4.6.1 si no 10.4.6.2. En el caso de la versión P1-ejb-cliente-remoto y P1-ejb-servidor-remoto se cambió la dirección de 10.4.6.2 a 10.4.6.1

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 en el PC host. Anote sus comentarios en la memoria.

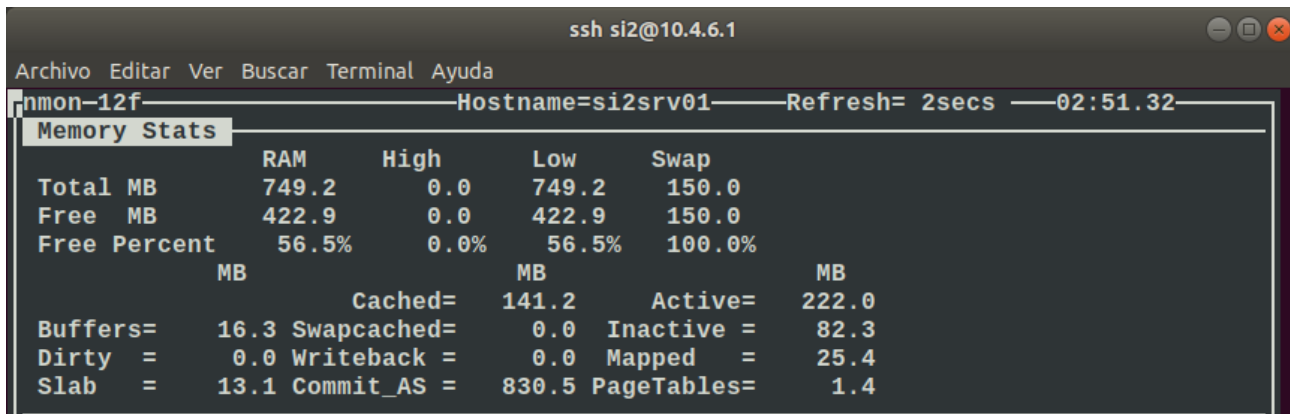
- **Resultados PC host:**

```
eps@eps ~$ free
              total        usado       libre   compartido búfer/caché   disponible
Memoria:    16402184    1920028    10550520    1650548    3931636    12540124
Swap:        7999484         0      7999484
```

```
nmon-16g-----[H for help]-----Hostname=eps-----Refresh= 2secs -----10:47.09-----
Memory and Swap
PageSize:4KB  RAM-Memory  Swap-Space      High-Memory      Low-Memory
Total (MB)    16017.8      7812.0          - not in use     - not in use
Free (MB)     10301.7      7812.0
Free Percent   64.3%       100.0%
Linux Kernel Internal Memory (MB)
                        Cached=    3625.8      Active=    1877.0
Buffers=      113.3  Swapcached=    0.0      Inactive =   3155.3
Dirty  =       0.4  Writeback =    0.0      Mapped  =   1514.7
Slab   =      187.8  Commit_AS =   9347.2  PageTables=    59.9
```

- Resultados PC1VM:

```
si2@si2srv01:~$ free
              total        used        free      shared    buffers     cached
Mem:          767168      329192      437976          0       16592      141004
-/+ buffers/cache:      171596      595572
Swap:        153592           0       153592
```



ssh si2@10.4.6.1

Archivo Editar Ver Buscar Terminal Ayuda

nmon-12f Hostname=si2srv01 Refresh= 2secs 02:51.32

Memory Stats

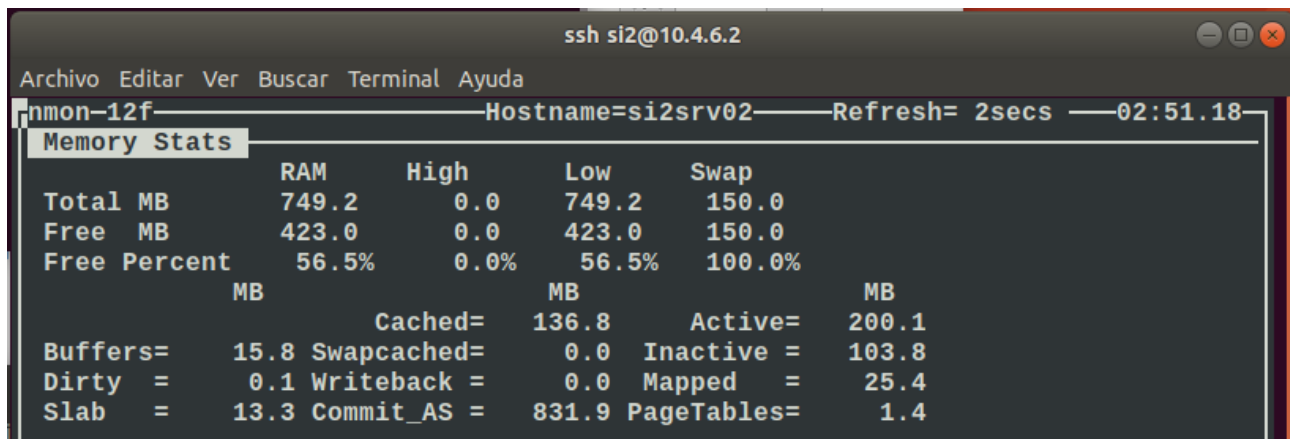
	RAM	High	Low	Swap
Total MB	749.2	0.0	749.2	150.0
Free MB	422.9	0.0	422.9	150.0
Free Percent	56.5%	0.0%	56.5%	100.0%

	MB	MB	MB
Cached=	141.2	Active=	222.0
Buffers=	16.3	Swapped=	0.0
Dirty =	0.0	Inactive =	82.3
Slab =	13.1	Mapped =	25.4
Commit_AS =	830.5	PageTables=	1.4

- Resultados PC2VM:

```
si2@si2srv02:~$ free
              total        used        free      shared    buffers     cached
Mem:          767168      332120      435048          0       16152      139916
-/+ buffers/cache:      176052      591116
Swap:        153592           0       153592
```



ssh si2@10.4.6.2

Archivo Editar Ver Buscar Terminal Ayuda

nmon-12f Hostname=si2srv02 Refresh= 2secs 02:51.18

Memory Stats

	RAM	High	Low	Swap
Total MB	749.2	0.0	749.2	150.0
Free MB	423.0	0.0	423.0	150.0
Free Percent	56.5%	0.0%	56.5%	100.0%

	MB	MB	MB
Cached=	136.8	Active=	200.1
Buffers=	15.8	Swapped=	0.0
Dirty =	0.1	Inactive =	103.8
Slab =	13.3	Mapped =	25.4
Commit_AS =	831.9	PageTables=	1.4

El PC host hace referencia a nuestra máquina particular que no son los ordenadores de los laboratorios y las pruebas para el **PC1VM** y **PC2VM** se han realizado en la misma máquina.

Respecto a los resultados que se muestran tras ejecutar cada comando, “free” que nos permite obtener el consumo de memoria indicando la memoria total, la consumida y la libre. Nuestra máquina host como se puede observar cuenta con 16Gb de memoria donde unos 2Gb están en uso y otros 10Gb están libres, en el caso de **PC1VM** y **PC2VM** en ambos observamos los mismos resultados, teniendo 768Mb de memoria total, con unos 300Mb en uso y 400Mb libres.

Mientras que el comando “nmon” también ofrece información sobre el consumo de la memoria pero más detallado.

**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.**

Realizaremos un pago a través de la página testbd.jsp para cada método y mostraremos una captura desde un gestor de BBDD demostrando que los pagos se han realizado correctamente.

- P1-base:



**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



**Pago con tarjeta**

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

idTransaccion: 1  
idComercio: 1  
importe: 15.9  
codRespuesta: 000  
idAutorizacion: 1

[Volver al comercio](#)

- P1-ws:



**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



**Pago con tarjeta**

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

idTransaccion: 2  
idComercio: 2  
importe: 9.63  
codRespuesta: 000  
idAutorizacion: 1

[Volver al comercio](#)

- P1-ejb:



**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



**Pago con tarjeta**

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

idTransaccion: 3  
idComercio: 1  
importe: 7.85  
codRespuesta: 000  
idAutorizacion: 3

[Volver al comercio](#)

- Consulta a la BBDD:

alumnodb@visa.10.4.6.1:5432 [8.4.10] SQL Editor - \*Untitled

```
1 select * from pago
2
3
```

#	idautorizacion	idtransaccion	codrespuesta	importe	idcomercio	numerotarjeta	fecha
1	1	1	000	15,9	1	8365 5667 6698 6481	24/03/22 03:08
2	2	2	000	9,63	2	7581 5513 5721 0259	24/03/22 03:10
3	3	3	000	7,85	1	3210 7991 3497 0393	24/03/22 03:13

### Ejercicio número 3:

Ejecuta 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;**

alumnodb@visa.10.4.6.1:5432 [8.4.10] SQL Editor

```
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.

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Min	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec
P1-base	1000	7	6	8	9	12	5	547	0,00%	134,0/sec	171,94	0,00
P1-ws	1000	48	45	54	61	79	38	665	0,00%	20,6/sec	26,56	0,00
P1-ejb	1000	13	13	17	18	23	9	711	0,00%	70,6/sec	92,11	0,00
Total	3000	23	13	49	52	70	5	711	0,00%	42,8/sec	55,26	0,00

Se puede apreciar como en la columna de Error el porcentaje de fallos es del 0.00% en todos los casos

Una vez que los resultados han sido satisfactorios:

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

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Min	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec
P1-base	1000	7	6	8	9	12	5	547	0,00%	134,0/sec	171,94	0,00
P1-ws	1000	48	45	54	61	79	38	665	0,00%	20,6/sec	26,56	0,00
P1-ejb	1000	13	13	17	18	23	9	711	0,00%	70,6/sec	92,11	0,00
Total	3000	23	13	49	52	70	5	711	0,00%	42,8/sec	55,26	0,00

- Salve el fichero server.log que se encuentra en la ruta glassfish/domains/domain1/logs de Glassfish y adjúntelo con la práctica.

Mediante el comando “scp si2@10.4.6.2:/opt/glassfish4/glassfish/domains/domain1/logs/server.log

ruta\_destino" pasamos el archivo de la máquina remota a la local.

- **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?**

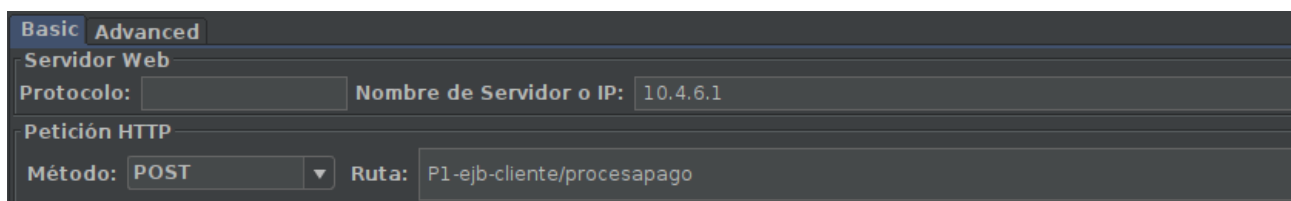
A partir de los datos que nos proporciona el Listener "Informe Agregado" concluimos con que la versión que ofrece un mejor resultado es P1-base. Basándonos en las explicaciones de cada columna que se indica en el apéndice 6.10, para decidir qué versión es mejor hemos consultado el valor de las columnas "Media" y "Rendimiento", donde P1-base obtiene mejores resultados frente a las otras versiones.

**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.**

Para aplicar los cambios que se comentan, en el test hacemos click derecho en los Threads P1-base y P1-ws y pinchamos en "Deshabilitar". Después modificamos la HTTP Request del Thread P1-ejb quedando tal que:



Los resultados obtenidos tras estos cambios son los siguientes:

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Min	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec
P1-ejb	1000	5	4	6	6	8	2	759	0,00%	183,8/sec	238,29	0,00
Total	1000	5	4	6	6	8	2	759	0,00%	183,8/sec	238,29	0,00

Se puede observar cómo respecto a la prueba anterior, han mejorado los propios resultados de este modelo y además, mejora los resultados del mejor modelo en la prueba anterior que era P1-base.

La razón por la que se obtienen estos resultados y por lo que se produce una mejora es porque se usa el cliente local en vez del remoto evitando tener que conectarse a él.

## Ejercicio número 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.xml3 . 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.**

Sobre el servidor Glassfish instalado en PC2VM realizamos los cambios que se comentan en el apéndice 8.5

- 1º: Utilizar la máquina virtual Java en modo servidor.

En nuestro caso se encontraba en modo cliente por lo que accediendo a "Configurations -> server-config -> JVM Settings -> Pestaña JVM Options", eliminamos la opción "-cliente" seleccionando dicha opción y pinchando en "Delete"

y añadimos “-server” tras pulsar en “Add JVM Option”.

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**

General Path Settings **JVM Options** Profiler

✓ New values successfully saved.

### JVM Options

Manage JVM options for the server. Values containing one or more spaces must be enclosed in double quotes ("value string").

Configuration Name: server-config

Options (31)

Select	Value
<input type="checkbox"/>	-Dorg.glassfish.additionalOSGiBundlesToStart=org.apache.felix.shell,org.apache.felix.gogo.runtime,org.apache.felix.gogo.commands
<input type="checkbox"/>	-Djavax.management.builder.initial=com.sun.enterprise.v3.admin.AppServerMBeanServerBuilder
<input type="checkbox"/>	-Djavax.net.ssl.trustStore=\${com.sun.aas.instanceRoot}/config/keystore.jks
<input type="checkbox"/>	<b>-server</b>
<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

-2º: Configurar los valores de memoria mínimo y máximo asignados a la máquina virtual java.

En valor máximo ya se encontraba configurado en 512MB, pero el mínimo no estaba establecido por lo que lo añadimos



pulsando en “Add JVM Option “ y escribiendo “-Xms512m”. Los cambios se aplican desde la misma pantalla anterior

The screenshot shows the GlassFish Server Open Source Edition web console. The left sidebar contains a tree view of the server configuration, with 'JVM Settings' selected under 'Configurations'. The main panel displays the 'JVM Options' configuration page for the 'server-config'. The page has tabs for 'General', 'Path Settings', 'JVM Options', and 'Profiler'. The 'JVM Options' tab is active, showing a list of 32 options. Two red arrows point to the '-Xmx512m' and '-Xms512m' options. A yellow banner at the top right says 'New values successfully'.

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
    - Security
    - System Properties
    - Thread Pools
    - Transaction Service
    - Virtual Servers
    - Web Container
- Update Tool

General Path Settings JVM Options Profiler

New values successfully

### JVM Options

Manage JVM options for the server. Values containing one or more spaces must be enclosed in double quotes ("value string").

Configuration Name: server-config

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.runtime
<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
<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}/jre/lib/ext\${path.separator}
<input type="checkbox"/>	-Djava.security.policy=\${com.sun.aas.instanceRoot}/config/server.policy
<input type="checkbox"/>	-Dgosh.args=--noInteractive
<input type="checkbox"/>	-Xms512m

-3º: Eliminar el despliegue y recarga automáticos de aplicaciones y activar la recopilación de JSPs.



← → ↻ No es seguro | https://10.4.6.2:4848/common/index.jsf

YouTube Twitch iCloud Gmail MOODLE DE... Horde :: Iniciar... GitHub Projects · Das... iLovePDF Free PDF, Vide

Home About...

User: admin | Role: domain1 | Server: 10.4.6.2

GlassFish™ Server Open Source Edition

Total # of available updates : 1 ⚠ Restart Required

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

Domain Attributes Applications Configuration Administrator Password Password Aliases Domain Logs

✓ New values successfully saved.

### Applications Configuration

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

[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 to the autodeploy directory.

**Auto Deploy Poll Interval:** 2 Seconds  
Frequency at which the autodeploy directory is checked for applications; interval does not affect amount of time to load.

**Auto Deploy Retry Timeout:** 4 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:** Full  
Type of deployment descriptor validation.

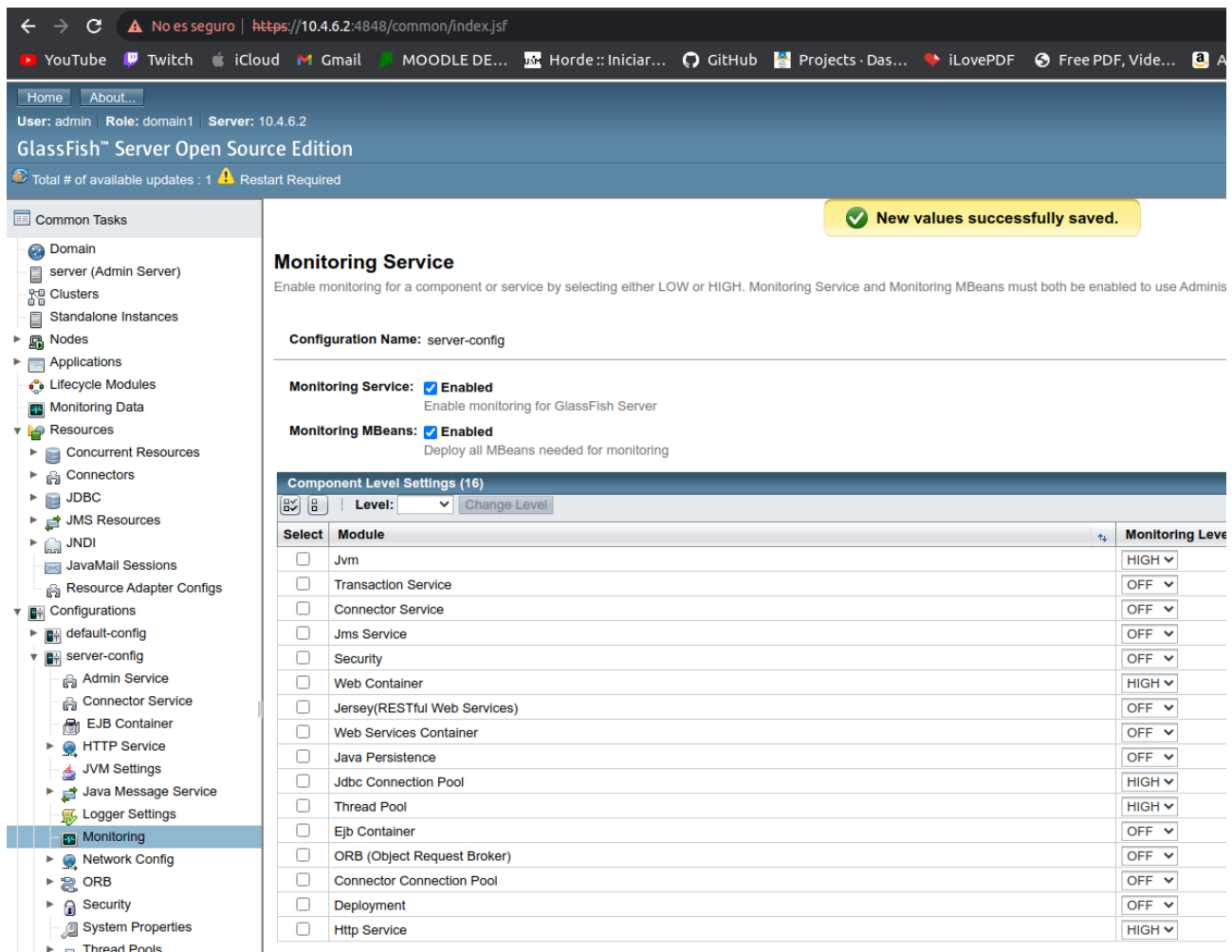
**Verifier:** ☐ Enabled  
Performs detailed verification before deployment.

**Precompile:** ☒ Enabled  
Precompiles JSPs resulting class files.

-4º: Activar un nivel de monitorización adecuado.

Vamos a la sección "Configurations -> server-config -> Monitoring" y activamos el modo de monitorización "HIGH" para los servicios Web Container, Thread Pool, JVM, JDBC Connection Pool, HTTP Service y el resto de la monitorización

se pone en "OFF".



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
    - Security
    - System Properties
    - Thread Pools

**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 Admins

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

Una vez aplicados todos los cambios, detenemos el servidor de aplicaciones y pasamos la nueva configuración del ordenador remoto al local con el comando scp "scp si2@10.4.6.2:/opt/glassfish4/glassfish/domains/domain1/config/domain.xml /home/eps/Descargas".

Para comprobar que los cambios se habían aplicado correctamente buscamos en el fichero traído del servidor remoto que módulos se encontraban con un nivel de monitorización HIGH encontrando la siguiente línea de código:

```
<module-monitoring-levels jvm="HIGH" jdbc-connection-pool="HIGH" thread-pool="HIGH" http-service="HIGH" web-container="HIGH"></module-monitoring-levels>
```

Se pueden observar como corresponde a los 5 módulos que se indicaron en el 4to objetivo.

**Revisar el script si2-monitor.sh e indicar los mandatos asadmin4 que debemos ejecutar en el PC host para averiguar los valores siguientes, mencionados en el Apéndice 1, del servidor PC1VM1:**

Revisando el script a partir de la siguiente línea:

```
asadmin --host $HOST --user $GFUSER --passwordfile $GFPASSFILE
```

Obtuvimos una idea de cómo ejecutar el comando asadmin y después solo hay que variar qué información queremos obtener concatenando con "." las diferentes pestañas del panel de administración por el que pasaremos para obtener la información que se quiera.

### 1. Max Queue Size del Servicio HTTP

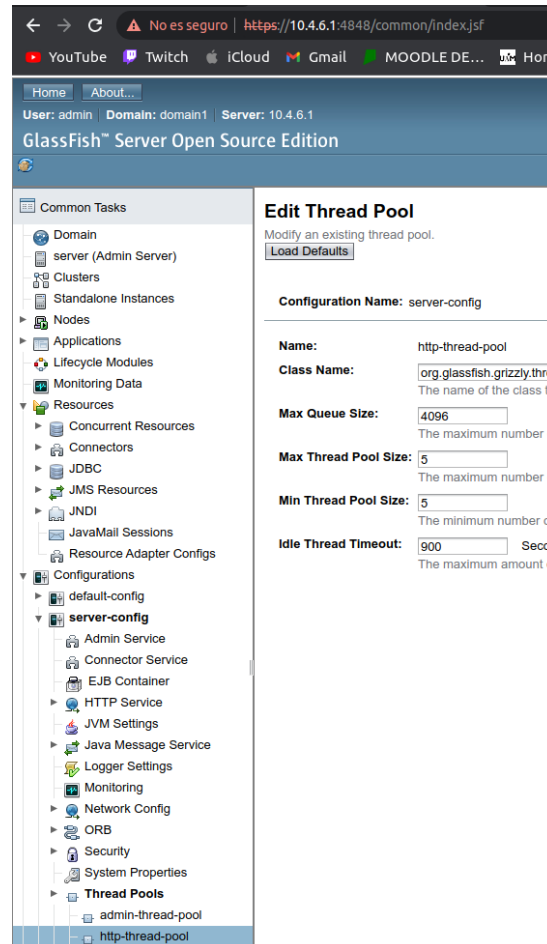
En el apéndice 13.4.1 vimos que la información relacionada con el tamaño del pool viene indicada en la ruta "Configurations->server-config->Thread Pools->http-thread-pool". Una vez sabido esto ejecutamos el siguiente comando, obteniendo que el tamaño máximo de la cola es de 4096:

Comando: `asadmin --host 10.4.6.2 --user admin --passwordfile ./passwordfile get configs.config.server-config.thread-pools.thread-pool.http-thread-pool.max-queue-size`

Salida:

```
eps@eps ~/Escritorio/Si2/P2 master ± asadmin --host 10.4.6.2 --user admin --passwordfile ./passwordfile get configs.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.
```

Se puede comprobar que es el valor correcto si accediendo manualmente desde el panel de administración y comprobamos que el valor coincide:



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

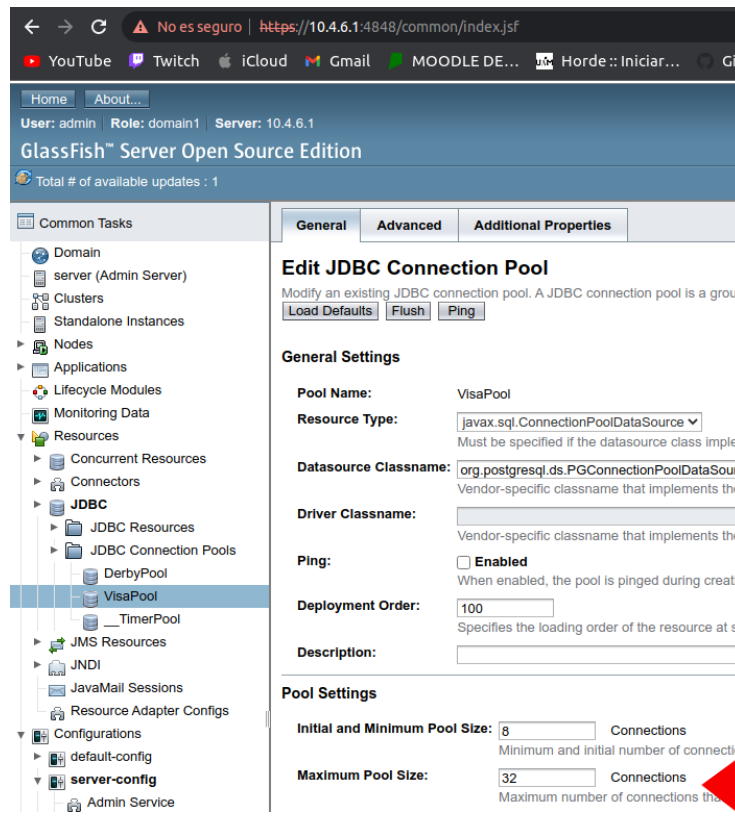
En el apéndice 13.4.1 vimos también que la información relacionada con el tamaño del pool viene indicada en la ruta “Configurations->server-config->Thread Pools->http-thread-pool”. Una vez sabido esto ejecutamos el siguiente comando, obteniendo que el tamaño máximo del pool es de 32 conexiones:

Comando: `asadmin --host 10.4.6.2 --user admin --passwordfile ./passwordfile get resources.jdbc-connection-pool.VisaPool.max-pool-size`

Salida:

```
eps@eps ~/Escritorio/Si2/P2 master ± asadmin --host 10.4.6.2 --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.
```

Se puede comprobar que es el valor correcto si accedemos manualmente desde el panel de administración y comprobamos que el valor coincide:



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

Comando: `asadmin --host 10.4.6.2 --user admin --passwordfile ./passwordfile monitor --type httplistener`

Salida:

```
eps@eps ~/Escritorio/Si2/P2 master ± asadmin --host 10.4.6.2 --user admin --passwordfile ./passwordfile monitor --type httplistener
ec  mt  pt   rc
3   2365 54.00 136
3   2365 54.00 136
3   2365 54.00 136
```

Ocurren 3 errores que se pueden observar en la columna "ec" que hace referencia a "error count"

## Ejercicio número 5:

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

Accedemos a las diferentes rutas del panel de administración y registramos los valores en la tabla de "Parámetros de configuración" del excel proporcionado por el equipo docente.

Parámetros de configuración		
Elemento	Parámetro	Valor
IVM Settings	Heap Máx. (MB)	512
IVM 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 número 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:***

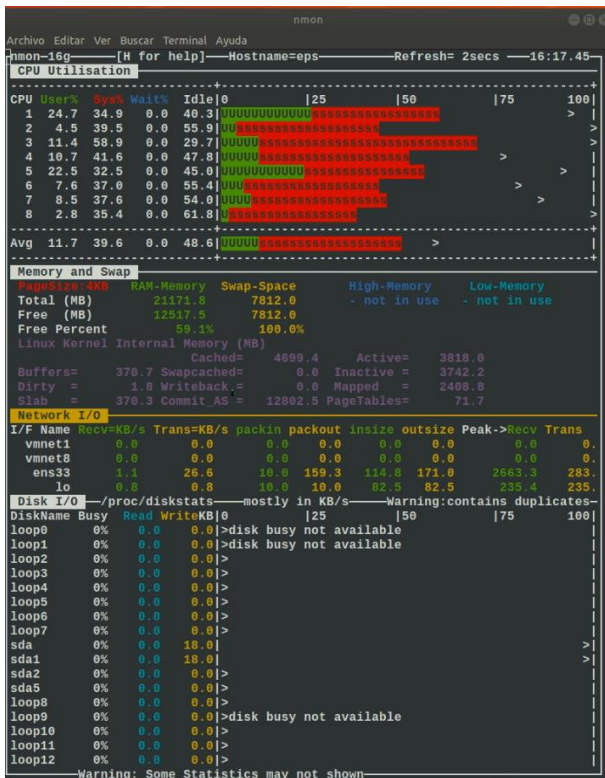
- ***A la vista de los resultados, ¿qué elemento de proceso le parece más costoso? ¿Red? ¿CPU? ¿Acceso a datos? En otras palabras, ¿cuál fue el elemento más utilizado durante la monitorización con nmon en un entorno virtual? (CPU, Memoria, disco ...)***

En primer lugar vamos a mostrar una serie de capturas para cada máquina, en la que se reflejan distintos momentos de la prueba (se realizó un vídeo y de él se recogen las diferentes capturas), mostrando una captura más al inicio de la prueba y otra durante la prueba. Además se mostrará una imagen de lo que se muestra al

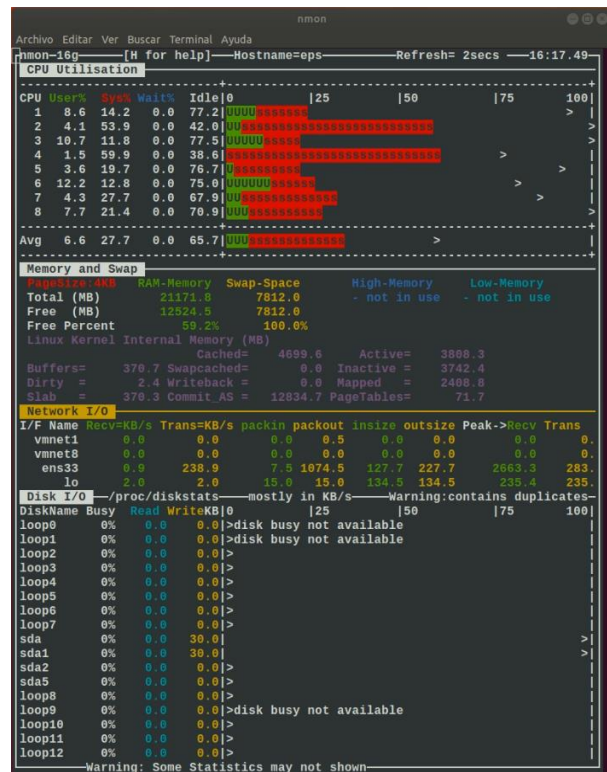


ejecutar el script “si2-monitor.sh”

- Resultados PC host:

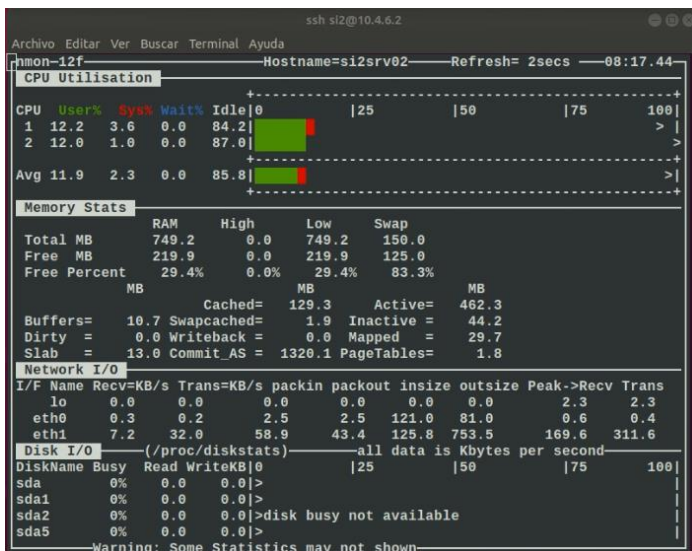


Situación inicial

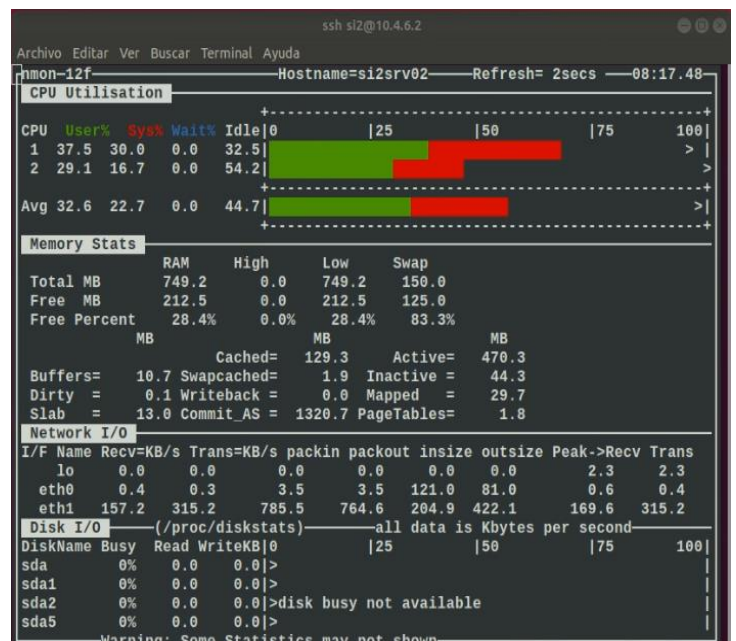


Situación intermedia

- Resultados PC2VM:



Situación inicial



Situación intermedia

Una vez visto el video y analizado, junto con las capturas que se indican más arriba, los elementos más costosos son la CPU y la red, pero en especial la CPU. Se puede apreciar como desde la situación inicial hasta la intermedia, el consumo de la CPU aumenta considerablemente, aumentando tanto el consumo de la CPU

de usuario que viene indicada en color verde como la del sistema que viene indicado en color rojo. Como es lógico, al realizar las consultas al servidor el consumo de red también aumenta.

Respecto al consumo de memoria y de disco, de memoria no hemos notado que se vea afectada aunque del disco si, aunque en las imágenes no se aprecie, revisando el vídeo en algún momento se escribía en disco y pese a que esto es un proceso computacionalmente costoso no consideramos que sea el elemento que más consume puesto que no se prolonga mucho en el tiempo.

A la vez que se visualizaban los recursos haciendo uso de la herramienta nmon, se ejecutó el script sobre la máquina 2 (PC2VM) y estos fue lo que se apreciaba:

#Muestra	numJDBCCount	numHTTPCount	numHTTPO
0	-1	0	0
1	-1	0	0
2	-1	0	0
3	-1	0	0
4	-1	0	0
5	0	1	0
6	0	1	0
7	0	1	0
8	0	1	0
9	0	1	0
10	0	1	0
11	0	1	0
12	0	1	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
20	0	0	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
36	0	0	0
37	0	0	0
38	0	0	0
39	0	0	0
40	0	0	0
41	0	0	0
42	0	0	0
43	0	0	0
44	0	0	0
45	0	0	0
46	0	0	0
47	0	0	0
48	0	0	0
49	0	0	0
50	0	0	0
51	0	0	0
52	0	0	0
53	0	0	0
54	0	0	0
55	0	0	0
56	0	0	0
57	0	0	0
58	0	0	0
TOT. MUESTRAS	59		
MEDIA:	-0.0847458	0.135593	0

En base a cómo está configurado el test de pruebas donde las consultas se hacen una a una utilizando un único hilo de ejecución (en este caso solo se uso el hilo P1-base ya que los otros dos se inhabilitaron) las peticiones se hacen una a una y por ello en cada muestra solo aparece que hay 1 petición HTTP. Consideramos que si no se hubieran inhabilitado los otros dos hilos, solo saldría una petición por muestra puesto que en el plan de pruebas se marcó que el grupo de hilos se lanzarán separadamente.

☒ **Lanza cada Grupo de Hilos separadamente (i.e. lanza un grupo antes de lanzar el siguiente)**

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

No es una situación realista puesto que el plan de pruebas simula a un único usuario realizando 1000 peticiones (una única petición pero 1000 veces) cuando lo más realista sería que varios usuarios realicen peticiones de forma concurrente o al menos no todo tan de seguido, porque este plan de pruebas consiste en que un usuario realiza cada petición una tras otra sin espera de por medio.

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

Como se comentó en el apéndice 7, previo a realizar el ejercicio 2, en el PC Host está siendo ejecutado todo, tanto JMeter como las dos máquinas remotas esto conlleva a que de por si la máquina Host tenga un mayor consumo de recursos a nivel de CPU y RAM principalmente. Una solución sería poder ejecutar las dos



máquinas remotas en diferentes equipos reduciendo así el consumo de recursos de la máquina Host.

## **Ejercicio número 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.**

Del mismo modo que en el ejercicio 1, que era preparar el script de pruebas siguiendo una serie de pasos indicados en el enunciado, no se van a incluir capturas de la configuración realizada ya que el script JMeter (P2-curvaProductividad.jmx) se entrega junto a la memoria.

## **Ejercicio número 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.**

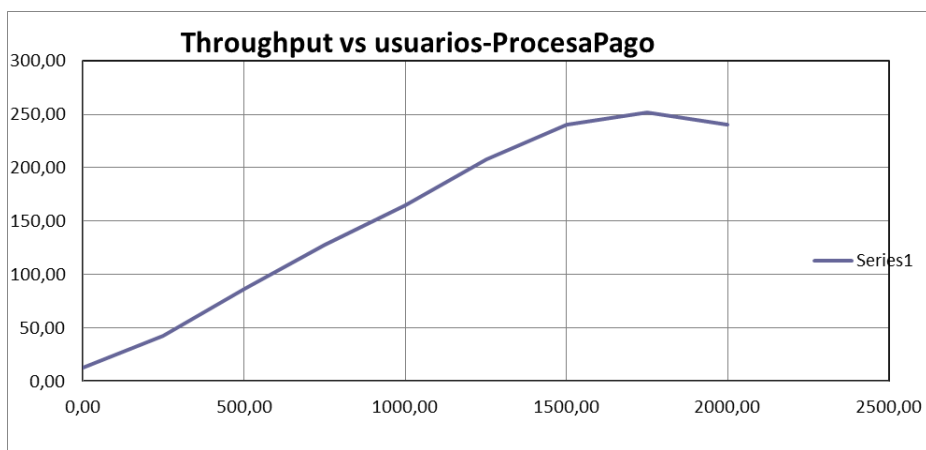
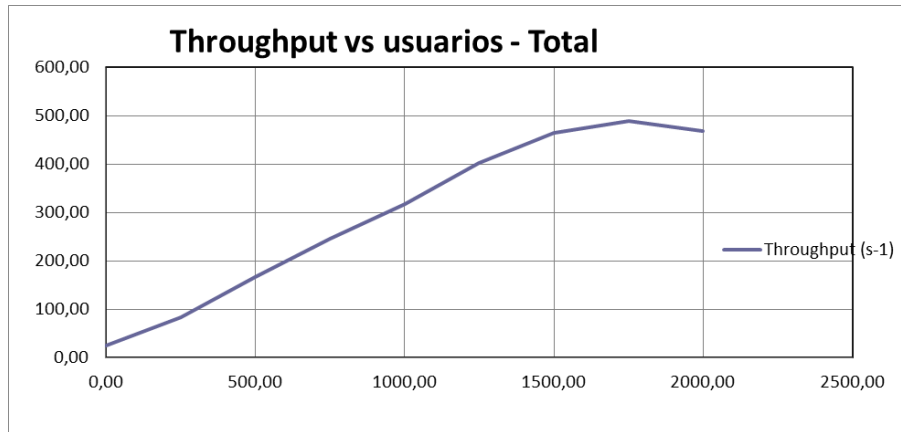
**Borrar los resultados de la ejecución anterior. En la barra de acción de JMeter, seleccionar Run -> Clear All.**

- **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.**

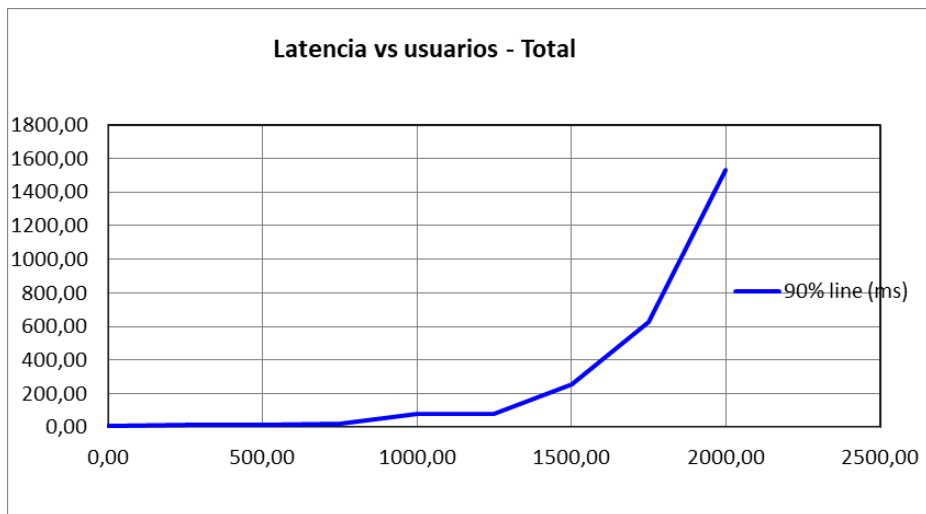
Se encuentra un archivo excel "SI2-P2-curvaProductividad.ods" donde se encuentran los datos recogidos de cada prueba junto con las gráficas de productividad y latencia, a continuación se mostrarán y se harán una

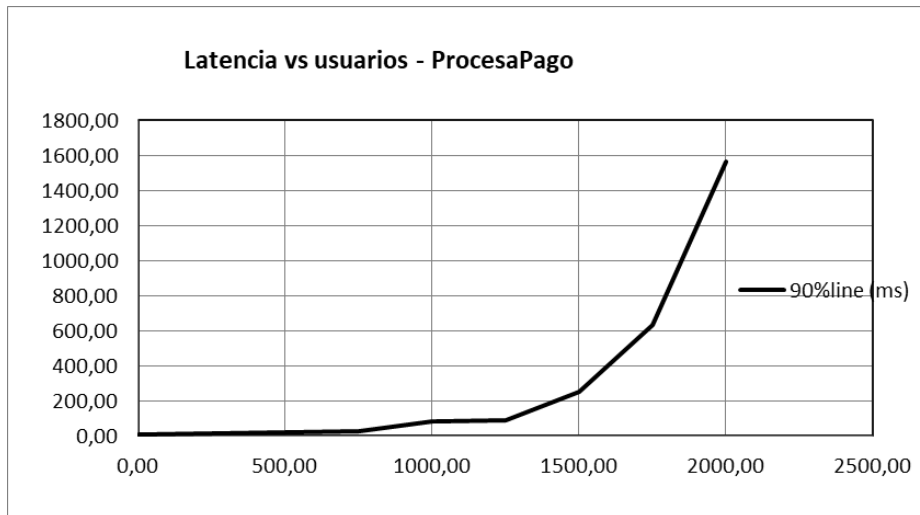
serie de aclaraciones ya que el análisis y las conclusiones se realizan en el siguiente ejercicio:

- **Productividad:**



- **Latencia:**





Hemos realizado las pruebas desde 1 usuario hasta 2000 usuarios donde el incremento ha sido de 250 en 250 como bien se indicaba en el enunciado. Los valores relacionados con la media, 90% line y productividad se han obtenido de los resultados del Aggregate Report. Por otro lado para saber el % de uso de la CPU se hizo uso del comando `vmstat -n 1 | (trap "INT; awk '{print; if(NR>2) cpu+=$13+$14;}END{print "MEDIA"; print "NR: ",NR,"CPU: ", cpu/(NR-2);}'");` recogiendo los resultados que este nos ofrece. Y, respecto a Visa Pool used Conns, HTTP Current Threads Busy y Conn queued se ejecutó el script sobre la IP del segundo PC (10.4.6.2).

Cabe recalcar que los resultados para cada prueba se incluyen en la carpeta "medidasEjercicio8" y en subcarpetas para cada número de usuarios involucrados en cada prueba. Por cada prueba se incluye una captura de los resultados del script "si2-monitor.sh", del comando "vmstat", también del comando "nmon -f" aunque en este caso sus valores no fueron utilizados y una imagen del Aggregate Report generada desde el propio JMeter.

## Ejercicio número 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.**

Se indicarán los resultados tanto para la gráfica de "Throughput vs usuarios - Total" como para "Throughput vs usuarios-ProcesaPago". Hay que recalcar que la zona de saturación es aquella en la que se ha alcanzado la capacidad máxima de proceso por lo que el sistema no es capaz de procesar más peticiones y donde la curva permanece constante.

- **Throughput vs usuarios - Total:**

En el caso de la productividad total a partir de 1250 usuarios el valor de la productividad permanece prácticamente constante no surgiendo variaciones que sean relevantes. En ese punto el valor de la productividad es de 401.40 s<sup>-1</sup>, pero el máximo es 488.3 s<sup>-1</sup> que ocurre para 1750 usuarios.

- **Throughput vs usuarios-ProcesaPago:**

En el caso de la productividad de ProcesaPago, ocurre una situación similar, el punto de saturación se encuentra a partir de los 1250 usuarios pero el valor es de 208.00 s<sup>-1</sup> y el valor máximo se obtiene también para 1750 usuarios pero con un valor de 251.90 s<sup>-1</sup>.

- **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.**

Acudiendo al apéndice 8.6 donde se comentaban los parámetros de configuración a registrar, concluimos que

para obtener el punto de saturación con un mayor número de usuarios podríamos modificar varios parámetros. Podríamos o bien incluir más peticiones simultáneas (Max Thread Pool Size) o poder tener un mayor número de conexiones en cola (Max Queue Size) aunque en esta las peticiones no se encuentren procesadas sí que las conexiones ya estarían establecidas y podría reducir tiempos.

Las razones por las que hemos elegido estas posibles soluciones provienen de los resultados que nos ofrece el script si1-monitor.sh ya que a partir de 1250 usuarios, el número de peticiones concurrentes está cerca del máximo (5) en media y el número de conexiones en cola se incrementa mucho.

- **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**

Se han realizado 3 pruebas independientes, la primera donde se aumenta el número de peticiones simultáneas, el número de peticiones en la cola y ambas a la vez. En la entrega se incluye otra carpeta llamada "medidasEjercicio9" donde se encuentran tres carpetas, una con los resultados cuando aumentamos el número de peticiones simultáneas, otra con los resultados cuando aumentamos el número máximo de conexiones en cola y otra donde se muestran las pruebas aplicando ambas mejoras. Para cada prueba se guardan los resultados del Aggregate Report, una captura de los resultados del script "si2-monitor.sh", y otra del comando "vmstat" para así comprobar que los cambios se han aplicado correctamente y poder comparar de forma más completa con las pruebas anteriores. A continuación, se comentarán los resultados obtenidos:

- **Respecto a Max Thread Pool Size:**

Se aumentó el valor de 5 a 15 el número máximo de peticiones simultáneas y se mantuvo el tamaño de la cola en 4096.

En este caso el rendimiento se ve afectado reduciéndose en vez de aumentando como pensábamos desde un principio. Si es verdad que el número de peticiones concurrentes en media aumenta un poco y disminuye el número de conexiones en cola y también el uso de la CPU ha aumentado de 47.81% a 52.24%.

- **Respecto a Max Queue Size:**

Se aumentó el valor de 4096 a 8192 el tamaño de la cola y se cambió de nuevo a 5 el número de peticiones simultáneas.

En este caso el número medio de conexiones en cola no se ha visto afectado demasiado y el % de uso de CPU ha aumentado de 47.81% a 59.78%, pero en este caso sí que el rendimiento ha aumentado de 401.4s-1 a 406.1s-1 en cuanto a rendimiento total y de 208.0s-1 a 211.5s-1 para procesaPago. No es un aumento considerable pero mejora.

- **Respecto a aplicar ambas mejoras a la vez:**

Se aumentó el valor de 5 a 15 el número máximo de peticiones simultáneas y se aumentó el valor de 4096 a 8192 el tamaño de la cola.

Esta situación es la que mejor rendimiento ha ofrecido, obteniendo 409.6s-1 respecto al rendimiento total y 212.7s-1 en cuanto al rendimiento procesaPago.