Departamento de
Automática y Computación
Automatika eta
Konputazio Saila

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

Tema 5 – Evaluación de prestaciones de un servicio Web

1. Objetivos

El objetivo principal de este trabajo es llevar a cabo un experimento de evaluación de un servicio (en concreto de un servicio web) e intentar aplicar las herramientas de análisis de prestaciones presentadas en otros temas.

2. Introducción

En esta práctica tomaremos el servidor web Apache y lo configuraremos en una o varias máquinas con el objetivo de evaluar sus prestaciones de cara al usuario. Para ello, se simplificará el funcionamiento del servidor mediante su configuración de forma que sea fácilmente analizable con los modelos presentados en clase y durante el tiempo de duración de esta práctica. A continuación se aprenderá a utilizar de forma sencilla la herramienta Apache JMeter con la que se realizarán peticiones web a dicho servidor. Finalmente, empleando JMeter y decidiendo los parámetros adecuados para el experimento obtendremos resultados respecto a la probabilidad de que el servidor esté congestionado y por lo tanto no pueda servir una página solicitada por el cliente.

Una vez obtenido un modelo para el servidor se abrirá la puerta a un estudio que nos permita decidir los parámetros de configuración del mismo en función del rendimiento que se desee obtener.

Por poner un ejemplo: el servidor web puede ser el de una empresa que ofrezca un servicio a través de Internet. Se desea decidir la configuración de dicho servidor de forma que se obtengan unos resultados de rendimiento buscados, por ejemplo que sirva al menos el 99.99% de las páginas que se le solicitan.

3. Máquina virtual y servidor web

El servidor web será httpd ejecutándose en una máquina virtual FreeBSD preparada para los efectos de este ejercicio.

Instalación de la máquina virtual

Se ha creado una máquina que ocupa varios centenares de megabytes. Debido a su tamaño no se va a poder guardar en los directorios HOME de los grupos de prácticas. Se recomienda utilizar el disco duro local de la máquina o su ordenador personal. En la web de la asignatura se ha indicado cómo obtener la máquina virtual. Se ha creado un directorio /opt/gprs/practica en cada máquina del laboratorio en el que tienen permiso de escritura. Descargue ahí dicha máquina virtual.

Importe la máquina virtual en VirtualBox o el sistema de vitualización que esté empleando. Recuerde indicar que la guarde en /opt/gprs/practica. La máquina virtual solo necesita ser accesible por el host así que por ejemplo en caso de usar VirtualBox puede emplear un adaptador de red “sólo-anfitrión”. Para más detalles sobre el proceso de importación puede recurrir al Apéndice A en este documento.

Departamento de
Automática y Computación
Automatika eta
Konputazio Saila

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

Esta máquina tiene creados los usuarios "root", "guest" y "tlm", todos sin contraseña. Acceda con el usuario **guest** que tiene preparado el directorio dir_httpd en su *HOME* de la máquina virtual, con todos los ficheros necesarios para lanzar Apache. Intentaremos hacer la mayor parte de la práctica con un usuario sin privilegios.

Normalmente Apache emplea ficheros de configuración que se encuentren en directorios que fueron especificados en su instalación y que suelen requerir privilegios de superusuario para ser modificados. Sin embargo, también se le puede indicar la localización de los ficheros al lanzarlo. Vamos a emplear esta segunda opción, lo cual nos permitirá tener los ficheros de configuración en el HOME de un usuario normal.

Los ficheros preparados para este ejercicio se encuentran en */home/guest/dir_httpd/*.

El directorio contiene la siguiente estructura de ficheros:

```
$ tree dir_httpd
dir_httpd
|-- htdocs
|   |-- phpinfo.php
|   |-- prueba.html
|   `-- randwait.php
`-- conf
    |-- httpd.conf
    |-- logs
    `-- run
4 directories, 4 files
```

El fichero *httpd.conf* contiene una configuración mínima de Apache, con las siguientes directivas:

```
# Run params
ServerRoot /usr/local
PidFile /home/guest/dir_httpd/conf/run/httpd.pid
Timeout 120
KeepAlive Off
MaxKeepAliveRequests 100
KeepAliveTimeout 15
UseCanonicalName Off
ErrorLog /home/guest/dir_httpd/conf/logs/error_log
LogLevel warn
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog /home/guest/dir_httpd/conf/logs/access_log common
AcceptFilter http none

ServerName 127.0.0.1

# Port
Listen 8087

# User & Group
User guest
```

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa

Departamento de
Automática y Computación
Automatika eta
Konputazio Saila

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

```
Group guest

# Modules
LoadModule authz_host_module libexec/apache22/mod_authz_host.so
LoadModule mime_module libexec/apache22/mod_mime.so
LoadModule mime_magic_module libexec/apache22/mod_mime_magic.so
LoadModule autoindex_module libexec/apache22/mod_autoindex.so
LoadModule dir_module libexec/apache22/mod_dir.so
LoadModule php5_module libexec/apache22/libphp5.so
LoadModule env_module libexec/apache22/mod_env.so
LoadModule log_config_module libexec/apache22/mod_log_config.so

# prefork MPM
# StartServers: number of server processes to start
# MinSpareServers: minimum number of server processes which are kept spare
# MaxSpareServers: maximum number of server processes which are kept spare
# MaxClients: maximum number of server processes allowed to start
# MaxRequestsPerChild: maximum number of requests a server process serves
<IfModule mpm_prefork_module>
    StartServers          1
    MinSpareServers       1
    MaxSpareServers       1
    MaxClients            1
    MaxRequestsPerChild   0
    LockFile /home/guest/dir_httpd/conf/logs/accept.lock
</IfModule>

# Maximum length of the queue of pending connections
ListenBacklog 1

# This forces the request to be treated as a HTTP/1.0 request even if it
was in a later dialect
SetEnv downgrade-1.0

# Documents directory
DocumentRoot /home/guest/dir_httpd/htdocs
<Directory />
     Options FollowSymLinks
     AllowOverride All
</Directory>
<Directory /home/guest/dir_httpd/htdocs>
     Options FollowSymLinks
     AllowOverride All
     Order allow,deny
     Allow from all
</Directory>
```

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

```
# Types
DefaultType text/plain
TypesConfig etc/apache22/mime.types
<IfModule mod_mime_magic.c>
     MIMEMagicFile etc/apache22/magic
</IfModule>
AddType application/x-compress .Z
AddType application/x-gzip .gz .tgz
AddType application/x-httpd-php .php
```

En la web de documentación del servidor web pueden encontrar descrito el objetivo y modo de uso de cada una de sus directivas. Sin embargo, a continuación resumiremos brevemente cómo emplear las pocas que se considerarán parámetros en este trabajo.

Lanzando Apache

Puede lanzar el servidor con la siguiente instrucción:

```
$ apachectl -f /home/guest/dir_httpd/conf/httpd.conf -k start
```

Para detenerlo habría que hacer:

```
$ apachectl -f /home/guest/dir_httpd/conf/httpd.conf -k stop
```

Lanzaremos Apache en un puerto no reservado, de forma que no haya necesidad de privilegios de superusuario. Esto se controla mediante la directiva *Listen*. Como se puede ver, en el fichero que se ha ofrecido se indica el puerto 8087 para el servidor.

Especificamos el directorio principal que contiene los ficheros ofrecidos por el servidor web mediante la directiva *DocumentRoot*, en este caso:

```
DocumentRoot /home/guest/dir_httpd/htdocs
```

Pueden lanzar un navegador web en el host/anfitrión (no en la máquina virtual) y acceder a una de las páginas de ejemplo que se les ha dejado con el siguiente URL:

http://VM_IP:8087/prueba.html

Donde VM_IP es la dirección IP del interfaz de la máquina virtual tal y como lo ve el host. Si no obtienen el fichero *$HOME/dir_httpd/htdocs/prueba.html* revisen los pasos anteriores.

A continuación comprobamos el correcto funcionamiento del módulo PHP accediendo a este script:

http://VM_IP:8087/phpinfo.php

Si no pueden ver el contenido del fichero */home/guest/dir_httpd/htdocs/phpinfo.php* entonces hay algo mal configurado; revisen los pasos anteriores. En caso de que el servidor esté correctamente configurado obtendrán una larga página HTML con información sobre la configuración y submódulos incluidos en el módulo PHP del servidor.

Configurando el número de procesos y peticiones simultáneas

Apache (con el módulo MPM prefork) emplea concurrencia de procesos para servir múltiples peticiones que se solapan en el tiempo. Es decir, lanza varios procesos y cada uno de ellos atiende a una petición hasta que termina de servirla, momento en que puede

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

atender a otra. Así, si tiene 10 procesos podrá estar sirviendo hasta 10 páginas simultáneamente a diferentes clientes. Podemos controlar la cantidad de procesos que lanza Apache mediante las directivas *StartServers, MinSpareServers, MaxSpareServers y MaxClients*. No vamos a entrar en mucho detalle sobre el uso de cada directiva y se recomienda para ello la documentación de Apache. Hemos indicado en el fichero de configuración de Apache de ejemplo:

```
StartServers                    1
MinSpareServers                 1
MaxSpareServers                 1
MaxClients                      1
```

con lo que Apache solo aceptará servir una petición a la vez (*MaxClients* = 1). Podríamos incrementar el número de peticiones simultáneas que sirve incrementando el valor de *MaxClients*, lo cual haría que Apache lanzara más procesos para atender dichas peticiones. Si alcanza este valor máximo de clientes simultáneos, nuevas peticiones (hasta que se complete una de las solicitudes) quedarán a la espera en cola. El sistema operativo aceptará la conexión TCP solicitada por el cliente web pero el servidor no atenderá a dicha petición hasta tener libre un proceso (de forma que mantenga el número de procesos ocupados como máximo en el valor de *MaxClients*). Se puede controlar el tamaño máximo de esta cola de espera mediante la directiva *ListenBacklog*. En el fichero de configuración que se ha entregado se ha configurado:

```
ListenBacklog 1
```

lo cual quiere decir que como mucho se mantendrá una conexión TCP aceptada en cola a la espera de que quede un servidor libre. Peticiones (segmentos TCP con bit de SYN activo al puerto del servidor Web) que lleguen mientras haya una aceptada en cola no completarán la conexión TCP. Sin embargo, según el sistema operativo puede que no cumpla exactamente con lo que indica esta directiva y acepte más conexiones de las que se esperaría con una cierta configuración.

Configurando la versión de HTTP

En este trabajo vamos a centrarnos en la versión 1.0 de HTTP, en la cual cada recurso que se solicita al servidor se hace en una conexión TCP independiente, cerrando el servidor la conexión al terminar de enviar el recurso. Dado que no hay forma (o no la hemos encontrado) de indicarle a JMeter que emplee la versión 1.0 del protocolo, se puede forzar esto en el servidor, y es lo que hemos hecho mediante la directiva:

```
SetEnv downgrade-1.0
```

Script randwait.php

Para este trabajo se ha preparado un pequeño script PHP el cual simulará el funcionamiento del servidor de aplicaciones de la empresa. Dicho servidor se supone que recibe peticiones de sus clientes, lleva a cabo la operación que se le ha solicitado, que le lleva un tiempo variable según la petición, y finalmente entrega una respuesta al cliente, cerrando a continuación la conexión. El script simula esto aceptando la petición de un navegador web, esperando durante un tiempo para simular el procesamiento de dicho servidor y finalizando a continuación la conexión tras entregar al navegador un pequeño texto diciendo que ha completado la operación.

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

El script acepta en la *query string* los siguientes argumentos:

- *distribution* : se indica aquí el tipo de distribución para la variable aleatoria del tiempo que tarda el servidor en procesar la petición. En estos momentos los valores posibles son:
  - *deterministic* : espera un tiempo predeterminado por el argumento *average*
  - *exponential* : espera un tiempo distribuido según una variable aleatoria exponencial de media el valor indicado en el argumento *average*
  - *uniform* : espera un tiempo distribuido según una variable aleatoria uniforme en [0, 2 x average]
- *average* : se indica con este argumento el tiempo medio que debe esperar el servidor a entregar el final de la respuesta. Está medido en microsegundos.

Ejemplos:

1. Esperar 10 segundos:

   http://VM_IP:8087/randwait.php?distribution=deterministic&average=10000000

2. Esperar un tiempo aleatorio según una distribución exponencial de media 100ms:

   http://VM_IP:8087/randwait.php?distribution=exponential&average=100000

4. Apache JMeter

JMeter es un programa Java diseñado para hacer pruebas de carga de servidores, por ejemplo servidores web. Utilizaremos este software en el host para evaluar experimentalmente las prestaciones de la configuración de Apache presentada en la parte anterior de esta práctica.

Desde un terminal puede ejecutar JMeter por ejemplo de la siguiente forma:

```
$ /usr/bin/jmeter
```

Puede que la última versión descargable en la web oficial sea más moderna que la instalada en el laboratorio. Es un programa Java que no requiere instalarse ni privilegios especiales así que si lo desea puede emplear en su lugar la última versión. Tenga en cuenta que puede haber cambios en los elementos disponibles y detalles de configuración. Lo mejor para cualquier duda es dirigirse a la documentación de esa versión.

Creando un plan de pruebas con un grupo de hilos

En la Figura 1 se puede ver la ventana de JMeter al lanzarlo. El elemento principal en JMeter es lo que se llama un “Plan de Pruebas” o “Test Plan”. El plan de pruebas contendrá todos los elementos que definirán cómo se hacen peticiones a un/os servidor/es.

Departamento de
Automática y Computación
Automatika eta
Konputazio Saila

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

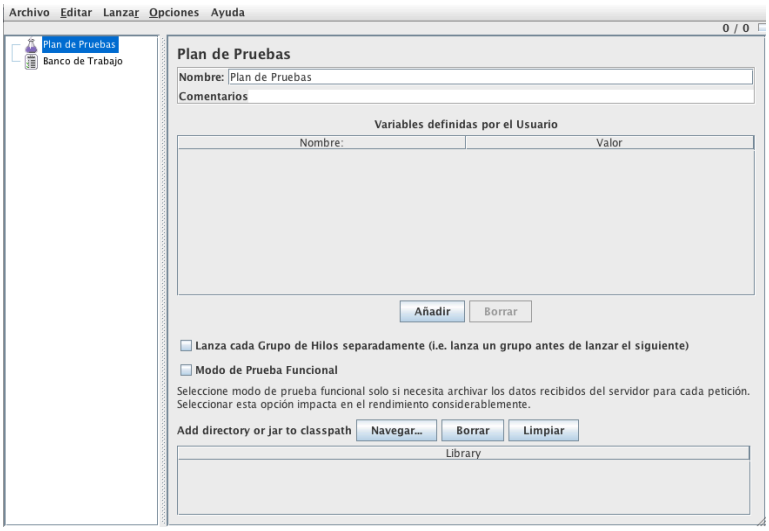

Figura 1 - Primera ventana de JMeter

Dentro de ese plan de pruebas añadiremos en primer lugar lo que se llama un "Grupo de hilos" o "Thread Group" mediante el menú que se despliega al pulsar el botón derecho del ratón sobre el plan de pruebas (Figura 2).

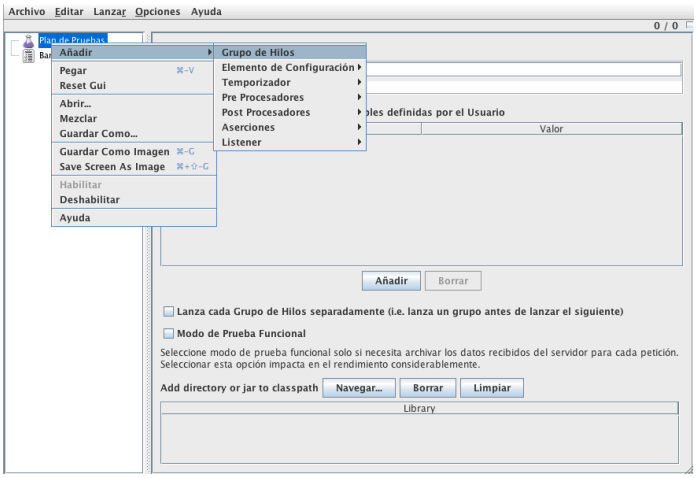

Figura 2 - Añadiendo un grupo de hilos

El grupo de hilos indica a JMeter que tendrá que lanzar un número de hilos ("threads") y qué deberán hacer estos hilos. Todos ellos van a hacer los mismo y será lo que se indique mediante elementos dentro del grupo de hilos. Lo que se pretende es que uno cualquiera de esos hilos emule el funcionamiento de un usuario o cliente web el cual estaría haciendo peticiones al servidor web, intercaladas con tiempos de descanso en los que se supone el usuario lee/procesa la página descargada antes de pedir la siguiente. Con el grupo de hilos controlaremos cuántos usuarios simultáneos está emulando JMeter (campo "Número de Hilos" en la Figura 3).

Departamento de
Automática y Computación
Automatika eta
Konputazio Saila

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

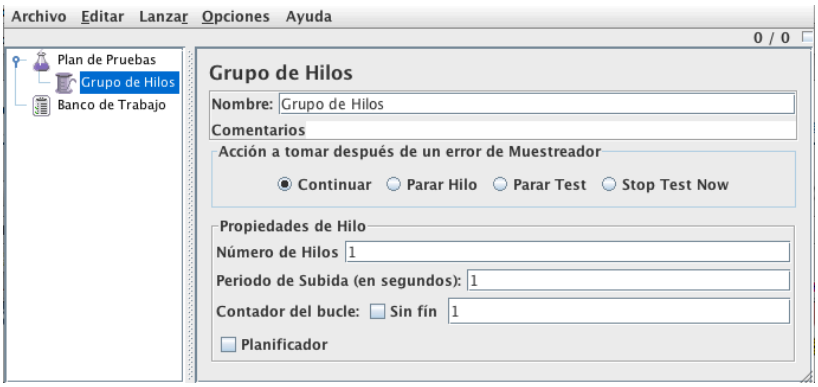

Figura 3 - Propiedades del grupo de hilos

El campo "Periodo de Subida" o "Ramp-up period" indica a JMeter que no lance todos los hilos inmediatamente al iniciar el experimento sino que espacie dicho arranque durante ese tiempo, de forma que sea más improbable que nos encontremos con que nada más empezar el experimento todos los clientes emulados coincidan a pedir una página.

Finalmente, el campo "Contador del bucle" o "Loop Count" indica cuántas peticiones (o iteraciones) debe hacer cada hilo. Si en cada iteración hay configurado por ejemplo una petición al servidor web seguida de un descanso y aquí indicamos 10, entonces cada hilo hará 10 peticiones intercaladas con otros tantos descansos.

En cualquier momento es recomendable seleccionar el plan de pruebas y guardarlo mediante la opción correspondiente del menú "Archivo".

Elementos básicos en el plan de pruebas

Los elementos descendientes del grupo de hilos que nos van a servir para indicar qué peticiones debe hacer un cliente son los "Muestreadores" o "Samplers", en concreto vamos a añadir un elemento de tipo "Petición HTTP" o "HTTP Request" (Figura 4).

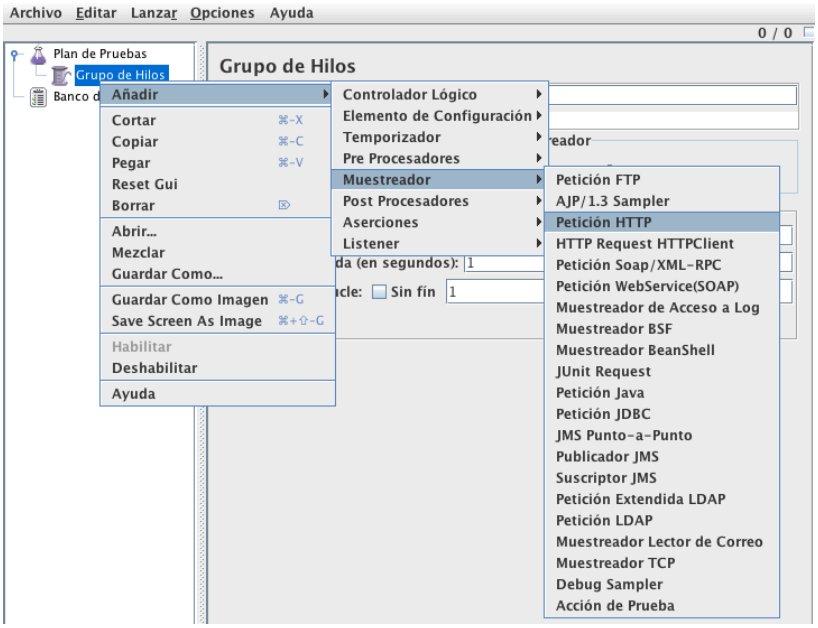

Figura 4 - Insertando elemento de petición HTTP

Departamento de
Automática y Computación
Automatika eta
Konputazio Saila

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

En la Figura 5 se ve que hemos indicado el servidor y puerto de nuestro servidor web (asumiendo que corremos Apache HTTPD y JMeter en la misma máquina) así como el camino (Path) al documento a pedir. Esto sería equivalente a pedir en el navegador web: http://localhost:8087/prueba.html

En nuestro caso, al estar corriendo Apache HTTPD en la máquina virtual habrá que indicar la dirección IP de la máquina virtual, una vez la hayamos arrancado (suponiendo que no hay un NAT entre la máquina virtual y el host).

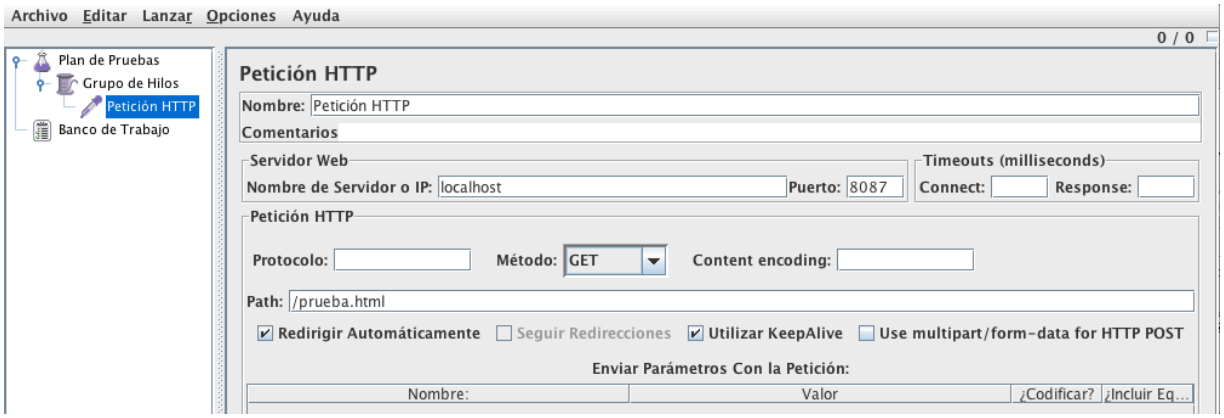

Figura 5 - Configurando la petición HTTP de prueba

Para ver los resultados de cada petición añadimos a continuación un "Listener". Para ello seleccionamos de nuevo el Grupo de Hilos en la columna izquierda y navegamos en el menú desplegable hasta añadir un "Listener" de tipo "Ver Árbol de Resultados" o "View Results Tree" (Figura 6).

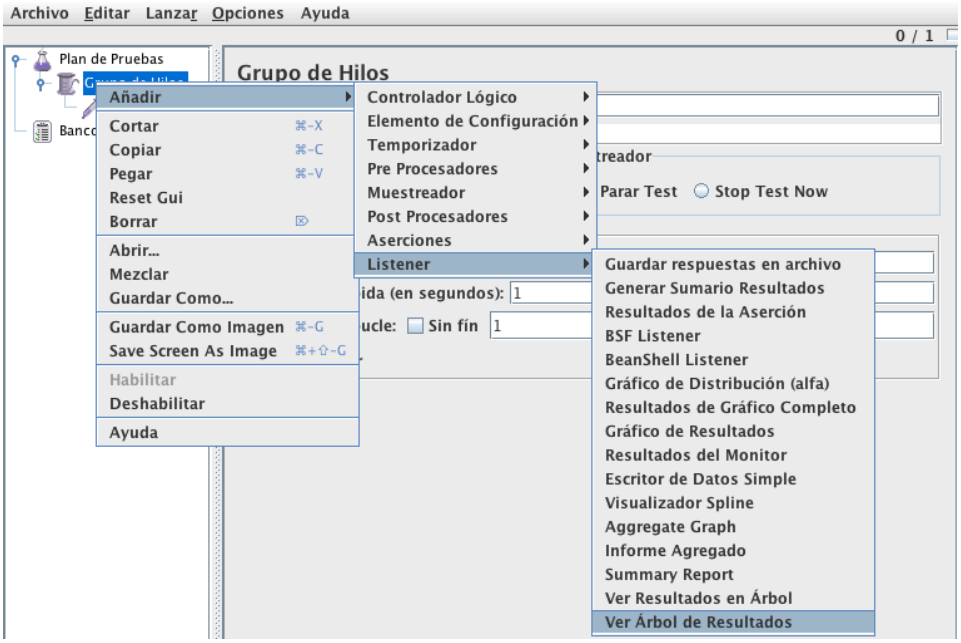

Figura 6 - Añadiendo un Listener

Podemos poner en marcha ahora el experimento mediante la opción "Arrancar" del menú "Lanzar" ("Run" -> "Start"). Como tenemos configurado que se lance un solo hilo y haga una sola iteración obtendremos algo similar a la Figura 7.

Departamento de
Automática y Computación
Automatika eta
Konputazio Saila

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

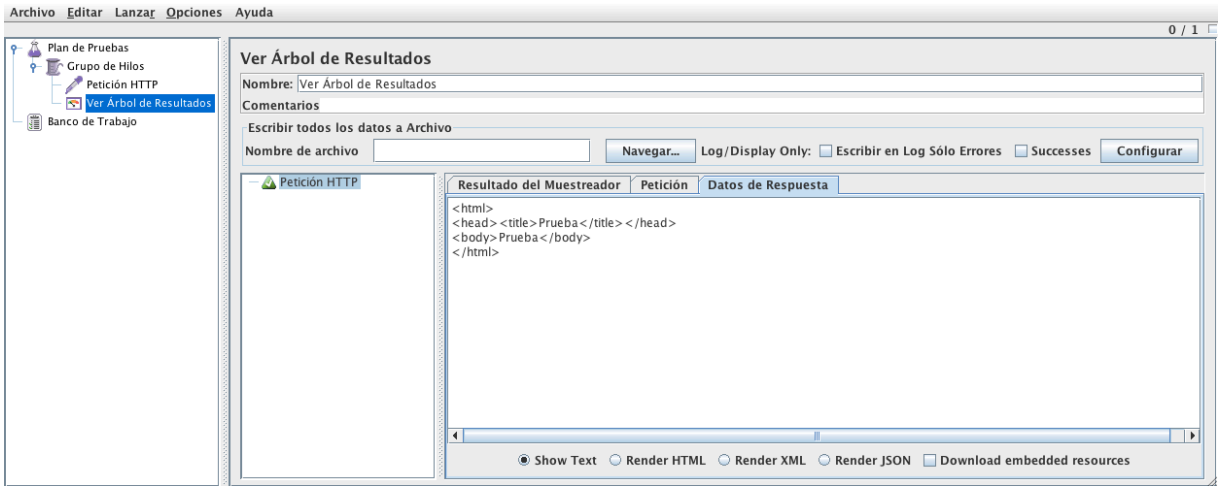

Figura 7 - Resultado de una petición

Vea en el fichero *access_log* de apache que ha recibido esta petición.

Si se indica un número mayor de 1 en el "Contador del bucle" del Grupo de Hilos, al terminar de recibir la respuesta a la petición HTTP y por lo tanto terminar la iteración pasaría el hilo a hacer una nueva petición. Esto no tiene en cuenta el tiempo que el usuario pasa procesando el recurso obtenido (o sea, leyendo la página web por ejemplo). Para tener en cuenta este punto incluiremos un "Temporizador" o "Timer", añadiéndolo al grupo de hilos mediante su menú desplegable. Seleccionamos por ejemplo un temporizador constante (Figura 8).

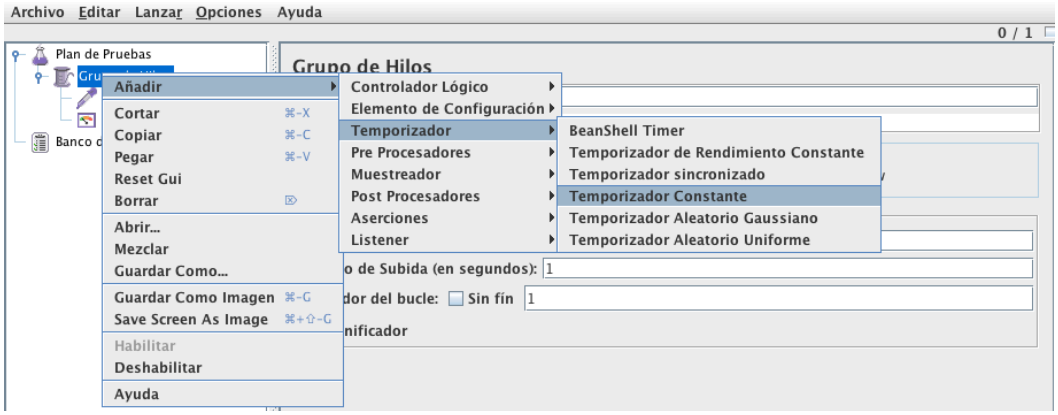

Figura 8 - Añadiendo un temporizador

En los parámetros de dicho temporizador se puede indicar la cantidad de tiempo (en milisegundos) que debe esperar el hilo (Figura 9).

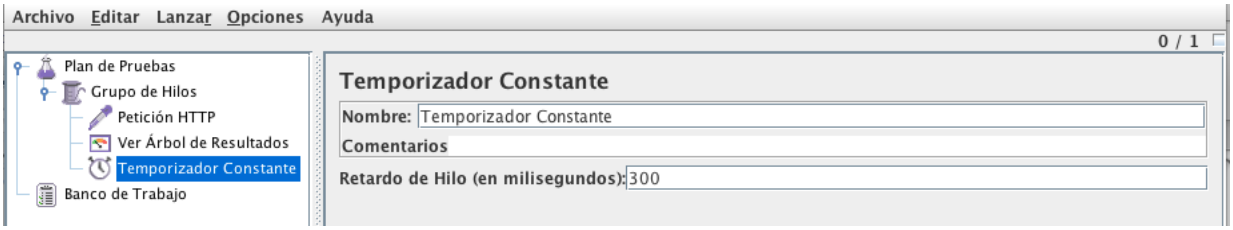

Figura 9 - Parámetro de retardo en temporizador constante

Departamento de
Automática y Computación
Automatika eta
Konputazio Saila

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

Hay que tener en cuenta que independientemente del orden en que estén colocados los elementos dentro del grupo de hilos, en cada iteración todos los temporizadores se procesan antes que los muestreadores. Es decir, en el ejemplo anterior, en cada iteración primero se bloquearía el hilo durante 300ms y a continuación haría la petición HTTP. Al terminar de recibir la respuesta del servidor habría concluido la iteración y si quedaban más iteraciones por llevar a cabo comenzaría de nuevo con el bloqueo de 300ms.

Aumente la pausa del temporizador a 2 segundos y el número de iteraciones a 5 (manteniendo un solo hilo) y compruebe que funciona como se ha descrito.

Con la configuración anterior aumente a 3 el número de hilos, con un periodo de subida de 3 segundos (de forma que el lanzamiento de los hilos estará espaciado en ese tiempo). Verifique con el log del servidor que el funcionamiento es el esperado.

Cambiamos a continuación que el elemento de petición HTTP en vez de pedir */prueba.html* solicite */randwait.php* . Para pasarle los parámetros al mismo podemos utilizar el botón de "Añadir" y rellenar "Nombre" y "Valor" de cada parámetro (Figura 10). También pueden poner el URL incluyendo los parámetros en al campo "Path" igual que lo harían en un navegador.

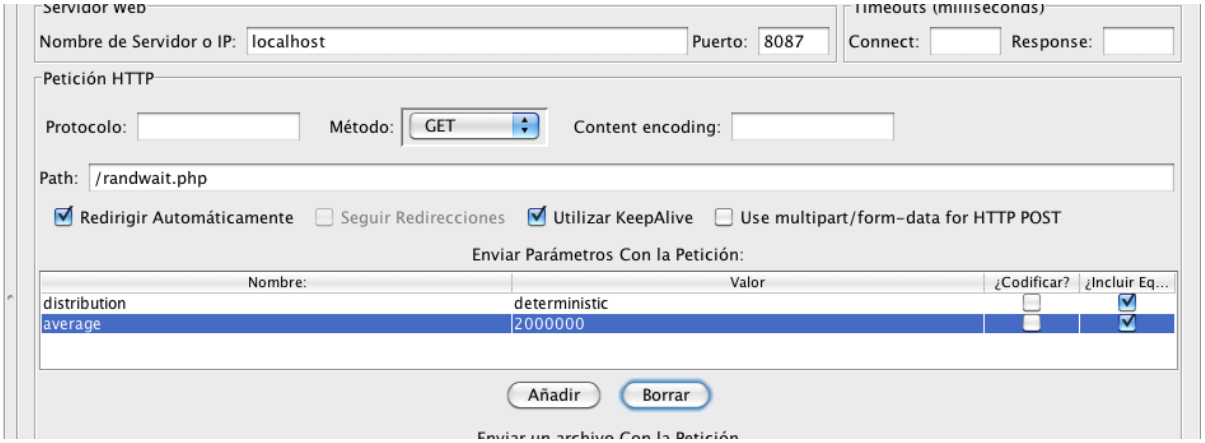

Figura 10 - Parámetros en la petición HTTP

Haga que JMeter pida a randwait.php una espera exponencial de media 2.5s.

Asociado al grupo de hilos puede haber varios muestreadores y varios temporizadores. Por ejemplo, podríamos añadir a continuación un temporizador de tipo aleatorio uniforme. Este temporizador lleva a que el hilo, en cada iteración, se bloquee un tiempo aleatorio que proviene de una distribución uniforme. De esta forma, ahora, en cada iteración, el hilo se bloquearía un tiempo determinista dado por el temporizador constante y un tiempo aleatorio uniforme dado por este temporizador; a continuación pasaría a ejecutar los muestreadores (en este ejemplo solo el de petición HTTP) y con ello completaría la iteración.

Cree un plan de pruebas en el que 2 hilos hagan peticiones a randwait.php indicando una espera del servidor en el servicio exponencial de media 1.5s con tiempos de espera del cliente entre peticiones según un timer uniforme entre 0 y 2s.

Emplee wireshark o tcpdump para calcular el tiempo que aproximadamente tarda en establecerse la conexión TCP entre el cliente (JMeter) y el servidor (HTTPD).

Departamento de
Automática y Computación
Automatika eta
Konputazio Saila

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

Captura a fichero

En prácticamente cualquier Listener se puede especificar un nombre de fichero en el que guardar los resultados. Todos los Listeners ofrecen las mismas opciones respecto a lo que guardar en ese fichero (botón "Configurar"). Por defecto guardan los resultados en formato XML, desmarque la opción "Guardar como XML" para que lo guarde en formato CSV (*Comma Separated Values*) (Figura 11).

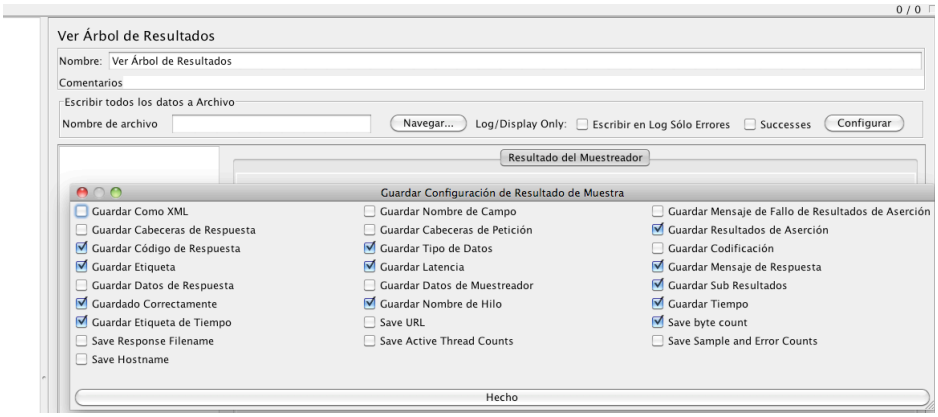

Figura 11 - Opciones de datos a fichero

Prueben con diferentes configuraciones el tipo de información que es capaz de guardar JMeter en ese fichero.

Peticiones rechazadas

Con la configuración básica de httpd que se ha hecho procesa una petición cada la vez y en teoría acepta y deja en cola como mucho otra. Sin embargo, como se ha comentado, según las implementaciones de TCP/IP puede variar cuántas conexiones acepta y deja a la espera de ser atendidas por la aplicación.

Empleando wireshark/tcpdump compruebe cómo se está comportando el sistema operativo de la máquina virtual respecto a dejar peticiones en cola, rechazarlas con Reset, no completar la conexión, etc. Por otro lado el cliente (el sistema operativo del host que está corriendo JMeter) podría hacer reintentos (retransmisiones de SYNs) en caso de no recibir respuesta.

Una vez que tenga claro el comportamiento tanto del servidor como del cliente, no vamos a cambiar el comportamiento del servidor (podríamos tocar parámetros del kernel pero no vamos a entrar a ello) y en el caso del cliente, dado que no hay pérdidas de paquetes en nuestro escenario, queremos que si no se establece la conexión con el primer SYN, que no se establezca a posteriori con retransmisiones. Es decir, si se envía el SYN y el servidor no contesta por tener completa la cola de conexiones aceptadas queremos que el cliente dé por fracasada esa petición. No queremos que por ejemplo 1 segundo después retransmita el SYN y sí la establezca porque el servidor esté entonces desocupado. Esto es una simplificación inicial para facilitar el análisis y posteriormente podemos retirarla. Para lograr este comportamiento puede intentar aprovechar el parámetro de Timeout que tiene el Sampler HTTP y hacer que caduque la petición antes de reintentos o emplear parámetros de configuración del mismo que evitan los reintentos (busque en la

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

documentación cómo indicarle propiedades a JMeter al lanzarlo y las propiedades del sampler http).

Cree un plan de pruebas en el que se produzcan rechazos de las peticiones y capture en fichero suficiente información como para calcular el porcentaje de peticiones rechazadas.

Tenga en cuenta que JMeter no pone a cero los contadores cada vez que se lanza un experimento, ni tampoco trunca el fichero donde captura los resultados, sino que los añade al final de él. Por tanto, use un fichero diferente cada experimento o borre el fichero antes de cada experimento.

Pruebe otros "Listeners" como por ejemplo el "Summary Report".

5. Experimento básico

El objetivo de esta parte es comprender las características y parámetros de funcionamiento del escenario concreto a analizar, así como el procedimiento que se pretende seguir. Veremos cómo generar un plan de pruebas que sea representativo de una población grande de usuarios del servidor y diseñaremos los experimentos de análisis de prestaciones a llevar a cabo. Nos centraremos en medir el número de peticiones rechazadas por el servidor en función de la carga.

Comportamiento de los usuarios

En las clases de teoría se han estudiado sistemas de colas en los que las llegadas seguían un proceso de Poisson. Existen varios buenos motivos para centrarse en este modelo, uno de los cuales es que es tratable matemáticamente para ofrecer resultados analíticos. Otro motivo es que el resultado de superponer un gran número de procesos independientes, con muy pocas condiciones adicionales sobre estos procesos, cumple que tiende a un proceso de Poisson al ir aumentando el número de procesos multiplexados.

En nuestro escenario, JMeter puede simular procesos ON-OFF donde el intervalo ON es aquel entre que realiza una petición y termina de obtener la respuesta y el intervalo OFF el tiempo de espera hasta hacer otra petición. Podemos suponer que aproximadamente durante todo ese periodo ON el servidor web tiene un proceso/hilo ocupado. La superposición de procesos de llegadas de este tipo se lleva a cabo indicando un número de hilos (Figura 12).

Departamento de
Automática y Computación
Automatika eta
Konputazio Saila

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

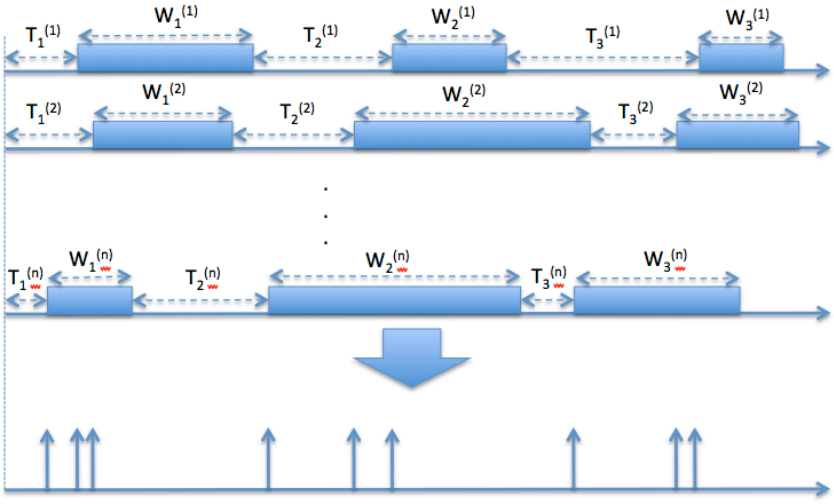

Figura 12 - Superposición de procesos de llegadas creados por hilos de JMeter

Construiremos a continuación un proceso de llegadas que se asemeje a un proceso de Poisson siguiendo el procedimiento descrito. Para ello comiencen un plan de pruebas de JMeter con un grupo de hilos. Incluyan en ese grupo de hilos un muestreador de petición HTTP y un timer de espera aleatoria uniforme. De esta forma, todos los $T_i^{(k)}$ de la Figura 12 serán variables aleatorias uniformes independientes e idénticamente distribuidas (i.i.d.). Haremos referencia a una cualquiera de estas variables aleatorias con T, siendo su media E[T]. El elemento de petición HTTP solicitará el recurso *randwait.php* indicando un tiempo de espera uniforme. Así, todos los $W_i^{(k)}$ de la Figura 12 serán variables aleatorias uniformes independientes e idénticamente distribuidas. Haremos referencia a una cualquiera de estas variables aleatorias con W, siendo su media E[W]. Llamaremos N_h al número de hilos.

Dada esta forma de crear el proceso, ¿cuál es el tiempo medio entre dos peticiones consecutivas del mismo hilo al servidor web? Llamaremos a este tiempo $E[Ti_h]$.

¿Cuál será la tasa media de peticiones por unidad de tiempo que hace un hilo cualquiera al servidor web? La llamaremos λ_h

¿Cuál será la tasa media total de peticiones que recibe el servidor web? La llamaremos λ_T

Dado que el tiempo de servicio que sufren las peticiones viene determinado por la variable aleatoria W, ¿cuál es la intensidad de tráfico que ataca al servidor? La llamaremos I.

Sistema simplificado

Hemos descrito la forma de crear un proceso aproximadamente poissoniano. En este proceso de creación especificamos un tiempo de servicio uniforme para el trabajo del servidor. El procedimiento no requería que dicho tiempo fuera uniforme sino que podría seguir cualquier distribución con momentos finitos. Lo mismo aplica al tiempo de OFF.

A partir de ahora usaremos para W una variable aleatoria exponencial. Eso implica que tenemos no solo un proceso de llegadas de Poisson sino también tiempos de servicio exponenciales i.i.d. Para el servidor web mantendremos la configuración presentada en partes anteriores de este trabajo, de forma que como mucho se atenderá una petición a la vez y configuraremos un ListenBacklog de 1 .

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

¿En notación de Kendall a qué tipo de sistema se ha reducido el problema?

Desviaciones del modelo

Una vez que el kernel del servidor acepta la conexión TCP, cuando el proceso de Apache esté libre tomará dicha conexión para atenderla. Desde ese punto el proceso estará ocupado y no podrá atender otras peticiones hasta que finalice esta tarea. El proceso debe cargar el script de PHP y ejecutar sus instrucciones, entre las que se encuentra la espera aleatoria según la distribución indicada. Es decir, el resultado final será que el proceso del servidor estará más tiempo ocupado que el tiempo de espera aleatoria, dado que a ese tiempo se añade el de ejecutar el código del script anterior y posterior a la pausa.

Hay que tener en cuenta también que estamos ejecutando Apache en una máquina virtual. Esto implica que es posible que el procesador físico tenga que hacer cambios de contexto entre el host y la máquina virtual, lo cual conlleva que el tiempo que esté ocupado el proceso de Apache sea aún mayor. Todo este aumento de tiempos será en general pequeño, pero si el valor de tiempo medio de espera que indicamos es también pequeño eso quiere decir que ese aumento de tiempos estará muchas veces en el mismo rango que el tiempo de espera, alterándolo en un porcentaje apreciable y con él la carga que está sufriendo el sistema frente a la que creemos. Así pues es conveniente que los tiempos de espera sean superiores a estos añadidos para que su efecto sea despreciable. Sin embargo, aumentar los tiempos de espera aumentará el tiempo que se tardará en llevar a cabo el experimento con un gran número de llegadas, por lo que tampoco querremos hacerlo muy grande.

Experimentación con un servidor

El objetivo es evaluar cómo depende el porcentaje de peticiones que no se sirven inmediatamente, de parámetros como la carga que sufre el sistema, el número de procesos que lanza el servidor, etc. En primer lugar nos planteamos evaluar simplemente la dependencia de dicho porcentaje con la intensidad de tráfico que ataca al servidor. El resultado podría ser una gráfica que nos muestre en el eje *x* la intensidad de tráfico y en el eje *y* el porcentaje (probabilidad) de pérdidas (ejemplo en la Figura 13).

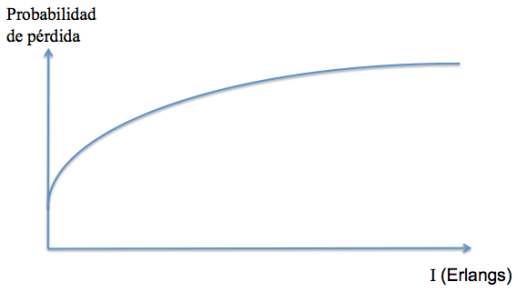

Figura 13 - Ejemplo de resultado

Cada punto en esta gráfica corresponderá a un experimento en el que habremos configurado unos parámetros en JMeter para que haga peticiones al servidor con la carga indicada y obtenido la probabilidad de pérdida medida como el porcentaje de peticiones que no fueron aceptadas por el servidor. Así pues, lo primero que necesitamos es decidir/calcular los valores de los parámetros de JMeter para crear una intensidad de

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

tráfico en concreto. Esto va a ser tan simple como el proceso inverso del que se ha pedido hacer con anterioridad en este documento.

El número de hilos se ha planteado como determinado por el número de usuarios que queremos emular. Querríamos probar el funcionamiento del sistema ante un gran número de usuarios, sin embargo, no es factible emplear JMeter con cifras de miles de usuarios/hilos y alta carga. Así pues, en vez de centrarnos en que los hilos emulen cada uno el comportamiento de un usuario nos centraremos en que la superposición de los hilos emule el multiplex de un número grande de los mismos. Para esto necesitamos que N sea lo suficientemente grande para que el proceso resultante "parezca" de Poisson, al menos a los efectos del rango de parámetros del sistema a evaluar. No vamos a dedicar tiempo en este trabajo a decidir un valor o valores para N sino que tomaremos directamente en torno a N=30 hilos.

Así pues, para cada intensidad de carga con la que desee experimentar, calcule unos valores razonables de E[W] y E[T]. Empleando en JMeter un Timer uniforme entre 0 y un valor T_{max} es sencillo decidir dicho valor T_{max} para que la media sea E[T].

Compruebe por ejemplo con el tráfico de red o con los logs que efectivamente el número medio de peticiones por unidad de tiempo que está haciendo el plan de pruebas se ajusta a lo deseado.

Queda por decidir el número de peticiones que hará cada hilo. Cuantas menos peticiones pongamos en el experimento menos tiempo tardará en llevarse a cabo, sin embargo, dado que estamos calculando el porcentaje de peticiones no atendidas necesitamos hacer suficientes peticiones para que el cálculo de dicho porcentaje tenga un grado de confianza suficiente. Por ejemplo, resulta evidente que si tenemos 30 hilos y a cada uno le pedimos que haga 10 peticiones tenemos un total de 300 peticiones, con las que es difícil por ejemplo estimar correctamente porcentajes de pérdidas del orden de 1%. Lo que se recomienda hacer es lanzar experimentos de muy larga duración y observar cómo a medida que se producen más llegadas se va estabilizando la estimación de la probabilidad de pérdida calculada con el porcentaje de pérdidas. Para ello en el plan de pruebas indicaremos que se guarde en fichero un log de las peticiones[1].

Calcule los parámetros necesarios para llevar a cabo los experimentos que le permitan crear una figura del estilo de la Figura 13.

Ensaye dichos parámetros con algunas intensidades de tráfico para verificar que los resultados se estabilizan en un tiempo razonable. Tenga en cuenta las limitaciones mencionadas con anterioridad. Pruebe a verificar que la carga que está sufriendo el sistema es la calculada teóricamente.

Para obtener el número de peticiones fracasadas tiene varias alternativas. Una de ellas sería en un Listener indicarle que guarde el código de respuesta de HTTP y en este escenario si no es 200 es que ha fracasado.

---

[1] No añada al plan de pruebas un *Listener* que por ejemplo guarde en memoria del programa todas las peticiones y sus respuestas pues en un experimento largo se distorsionarán las medidas por el tiempo que consumirá JMeter en mantener dicha lista en memoria.

Departamento de
Automática y Computación
Automatika eta
Konputazio Saila

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

Resultados (40%)

En el anterior apartado se han obtenido los parámetros para realizar un estudio del comportamiento de Apache con la configuración descrita previamente. Se ha supuesto que esta configuración con estos parámetros de tráfico que lo atacan se puede comparar a un modelo de teoría de colas de los estudiados teóricamente, pero habrá que comprobarlo.

Compare los resultados experimentales de probabilidad de fallo frente a intensidad de carga comparándolos con algún modelo teórico.

Aumentando la cola (30%)

Hasta ahora Apache estaba configurado para guardar una petición en cola, vamos a probar a aumentar este valor.

Aumente el valor de la cola hasta 3. ¿En notación de Kendall, ahora a qué tipo de sistema se ha reducido el problema? Realice otra vez los mismo experimentos y obtenga las probabilidades de pérdidas.

Si tiene tiempo pruebe con otros tamaños de cola.

Aumentando el número de procesos (30%)

Para concluir pruebe a cambiar el número de peticiones simultáneas que puede estar procesando el servidor Apache aumentándolo a 2 o más y repita el mismo tipo de experimentos.

6. Entregables

Prepare un documento donde presente la metodología seguida, los resultados de las pruebas intermedias, los resultados que ha obtenido con 1 o más servidores y diferentes tamaños de cola, comparando los resultados experimentales con los teóricos y analizando y discutiendo estos resultados.

Apéndice A: Importación de la máquina virtual en los ordenadores de los laboratorios de telemática

En primer lugar debe descarar el fichero .ova en el directorio /opt/gprs/practica/. Si lo descarga en su HOME es muy probable que no le quepa y además la descarga será más lenta (su directorio HOME se encuentra en un disco en red).

A continuación lance VirtualBox y seleccione la opción “Import Appliance…” del menú “File” y a continuación el fichero .ova que ha descargado (Figura 14).

Departamento de
Automática y Computación
Automatika eta
Konputazio Saila

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

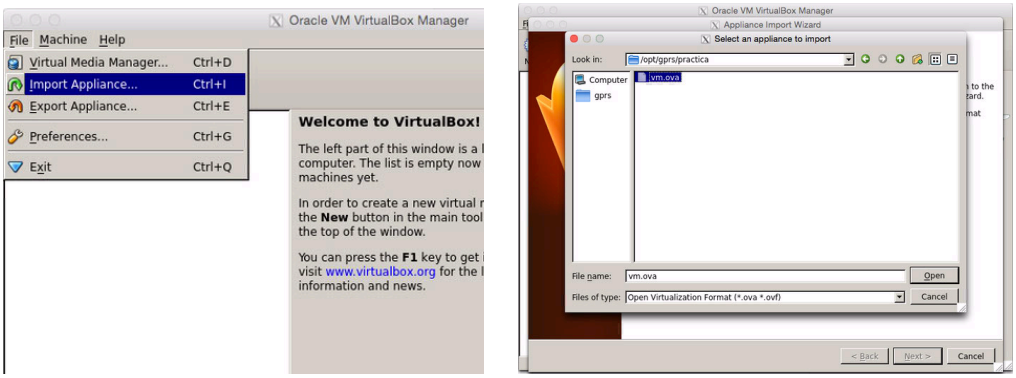

Figura 14 - Opción de importación

En las opciones de importación recuerde modificar el directorio donde guardará VirtualBox la máquina virtual una vez importada. Debe ser de nuevo en /opt/gprs/practica/ para que no haya problemas de espacio (Figura 15).

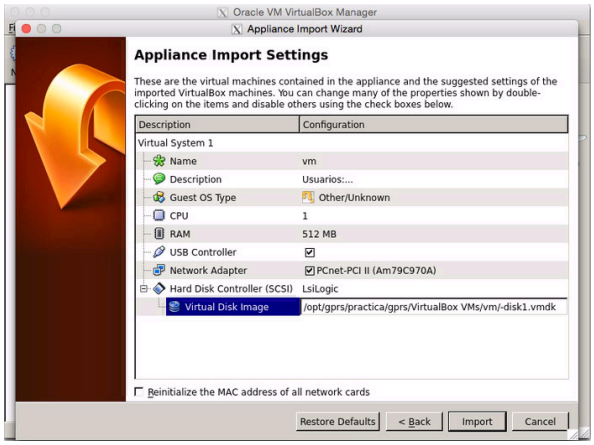

Figura 15 - Localización del disco

Una vez importada la máquina virtual (o antes de importarla) verifique que tiene creado el interfaz virtual vboxnet0, el cual servirá para comunicar al host (el sistema operativo Linux del PC) con el guest (el sistema operativo FreeBSD en la máquina virtual) (Figura 16).

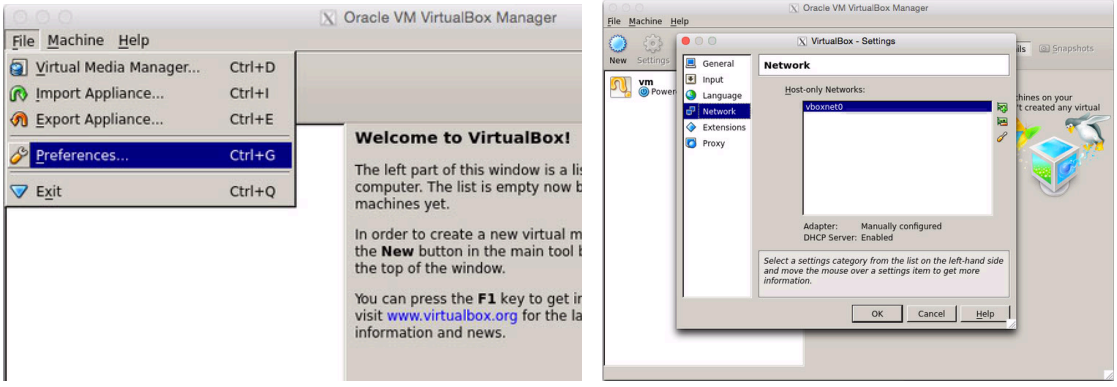

Figura 16 - Creación de interfaz vboxnet0

Departamento de
Automática y Computación
Automatika eta
Konputazio Saila

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

Si no lo tenía creado puede tener que configurar la máquina virtual para que lo emplee. Esto se haría en la configuración de la máquina virtual, en la sección "Network" (Figura 17).

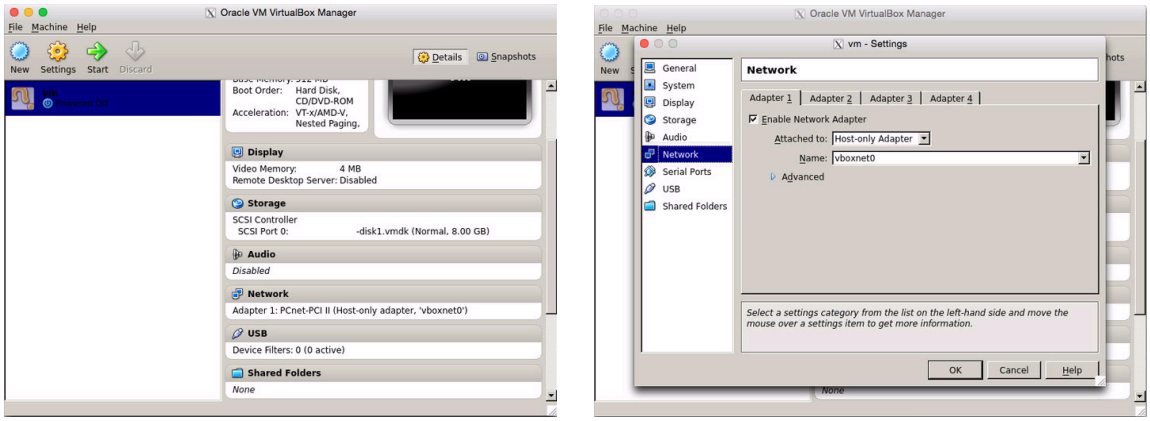

Figura 17 - Preferencias de red de la máquina virtual

Finalmente, una vez que arranque la máquina virtual podrá ver que obtiene una dirección por DHCP que le asigna la configuración hecha en VirtualBox para el interfaz vboxnet0. En este caso la dirección asignada ha sido 192.168.1.101 (Figura 18).

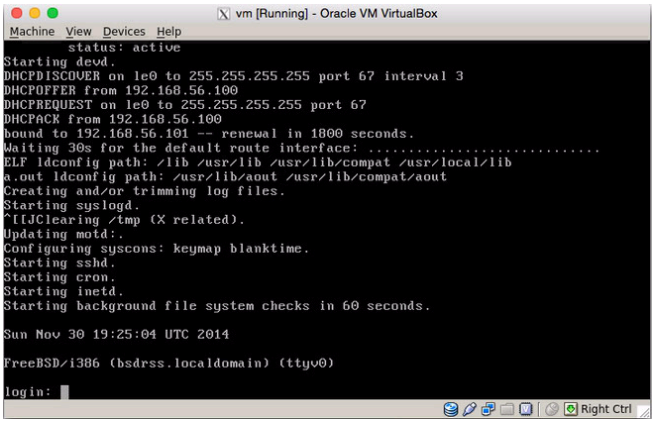

Figura 18 - Final del arranque de la máquina virtual

Puede emplear la máquina virtual directamente en la consola o puede acceder a ella mediante SSH (tiene un servidor de ssh activo) con el comando:

```
ssh guest@192.168.56.101
```