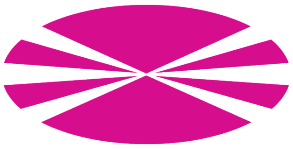

UNIVERSIDADE DA CORUÑA

Ferramentas de desarrollo
Informe
Pruebas de carga

Nombre	Email
Alejandro Fernández Garcia	alejandro.fernandez@udc.es
Alejandro Fortes Lopes	alejandro.fortes.lopes@udc.es
Elias Grande Casedas	elias.grande@udc.es

Índice

1. Consideraciones previas

Las pruebas de carga se han realizado en un portátil con la siguiente especificación:

- Procesador intel Core 2 Duo CPU P8700 a 2,53GHz
- Memoria RAM: 4GB
- Ubuntu 12.04

2. Benerator

Con el *Benerator* se han generado datos en la bd para que cuando se realizasen las peticiones las respuestas de estas peticiones no fueran vacías y simulase el correcto funcionamiento. Se ha generado valores para las tablas que eran estrictamente necesarias debido a que tuvieran una relación con *File* o *Album*.

Tabla	**Número de inserciones**
usuario	50
like_dislike	6000
album	1000
archivo	20000
comment	100000
album_tag	200000
file_tag	200000

Con lo de número de inserciones se refiere al número de elementos creados en cada tabla.

```
</setup>

<bean id="idGen" spec="new IncrementGenerator(1)" />
<bean id="dtGen" spec="new DateTimeGenerator()">
   <property name='minDate' value='2013-06-01' />
   <property name='maxDate' value='2013-12-01' />
</bean>
<setup>
   <generate type="album" count="1000" consumer="db">
      <variable name="person" generator="PersonGenerator" />
      <id name="id" generator="idGen" />
      <attribute name="name"
         script="person.givenName + ' ' + person.familyName" converter="ToLowerCaseConverter,
   UniqueStringConverter" />
      <reference name="user_id" source="db" targetType="usuario"
         cyclic="true" distribution="random" />
      <attribute name="date" type="timestamp" generator="dtGen" />
      <attribute name="privacy_level" constant="PUBLIC" />
      <reference name="like_dislike_id" source="db" targetType="like_dislike"
         cyclic="true" distribution="random" />
   </generate>
</setup>
```

Figura 1: Ejemplo de creación de datos de la tabla Album.

3. JMeter

Para simular el uso del servicio *REST* por un número elevado de usuarios, se ha utilizado JMeter con el que se han lanzado peticiones *HTTP* hacía el servicio *REST*.

Para ello se ha creado un plan de pruebas que ejecuta 100 hilos, sin descanso entre ellos.

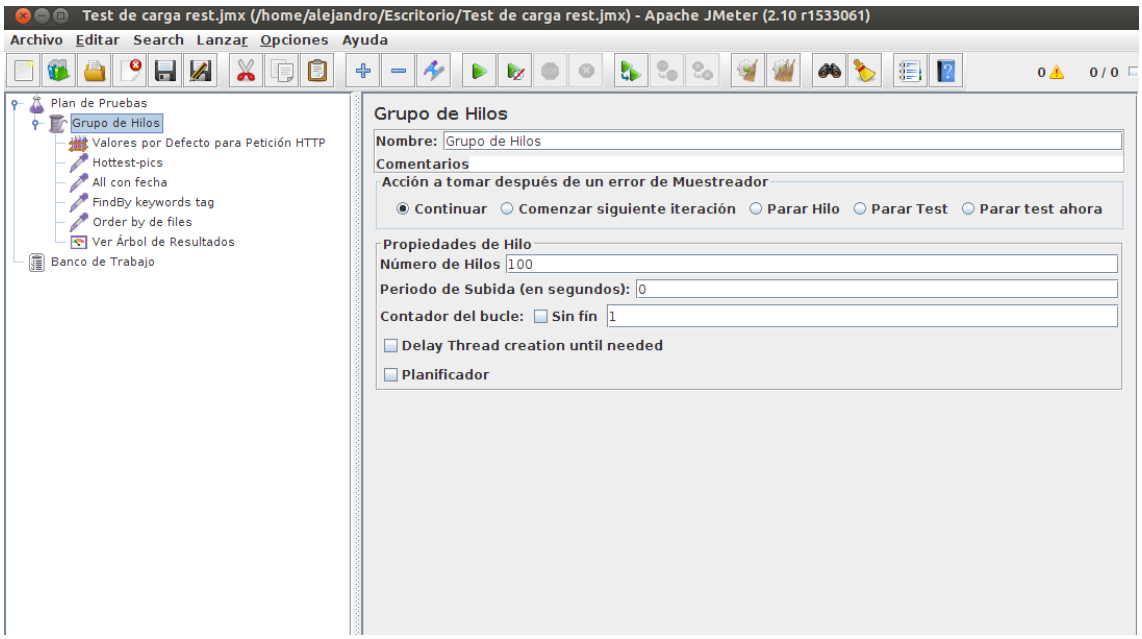

Figura 2: Configuración de los hilos en JMeter.

Como se puede apreciar en la imagen se han creado varias peticiones *HTTP* para probar distintos casos, intentando simular así una ejecución real por un número de 100 usuarios accediendo al servicio *REST* de forma concurrente.

Las peticiones son las siguientes:

- **Hottest-pics:** Consulta que devuelve los *Hottest-pics*.
- **All con fecha:** Consulta que devuelve tanto los *Files* como los *Album* que se crearon entre 2 fechas.
- **Findby keywords tag:** Consulta que devuelve los *Files* que contienen las palabras claves en sus tags.
- **Order by de files:** Consulta que devuelve los álbumes ordenados por el parámetro *like*.

El otro elemento que se ve en la imagen, *Ver Árbol de Resultados* es una vista que permite observar las respuestas a las distintas peticiones.

4. Java Mission Control

Se ha escogido el *JMC* para realizar un seguimiento de la memoria y el estado de la aplicación, durante la ejecución de la misma usando *JMeter*.

El consumo de memoria o cpu por parte del sistema cuando no se le realiza ninguna petición al *Tomcat* es prácticamente nulo.

Por el contrario, una vez lanzado el *JMeter* se ve un aumento de consumo tanto de la CPU viendo que prácticamente todo el consumo procede de la *JVM*.

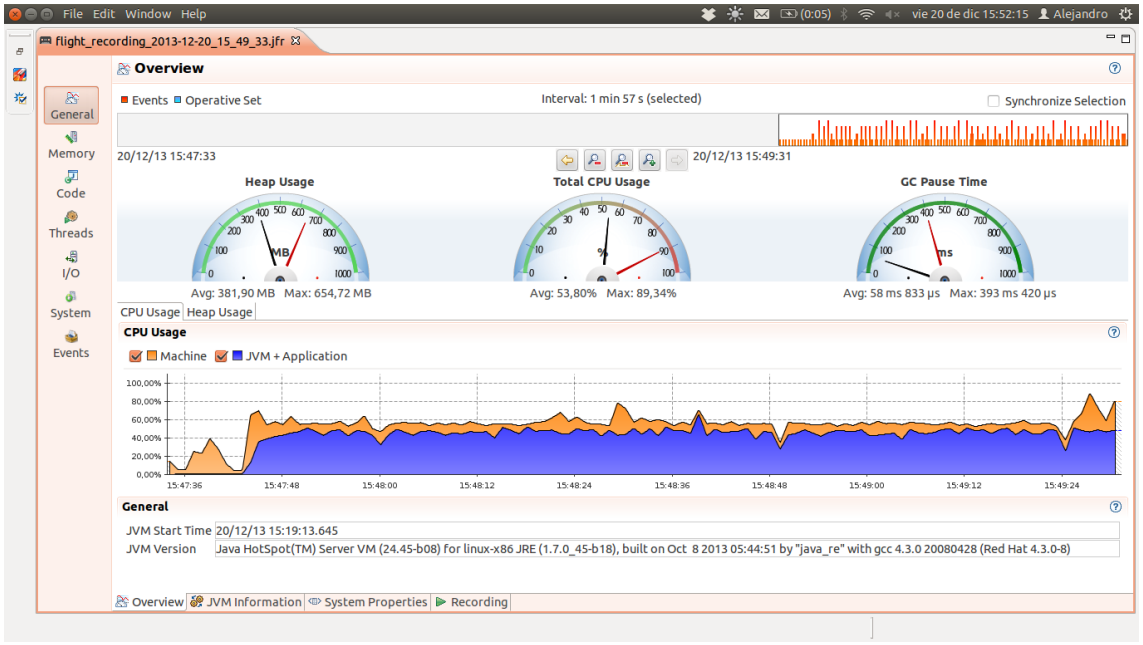

Figura 3: Gráfico de consumo de CPU.

Respecto al propio consumo dentro de la *JVM*, se puede ver (en la siguiente imagen) como durante el tiempo que se realizan las peticiones el consumo del *HEAP* se dispara. En esta imagen también se puede apreciar las veces que el *Garbage Collector* libera la memoria, especialmente entre el 15:48:32 y el 15:48:43 donde se puede ver como el *Garbage Collector* esta 400 ms para limpiar el *HEAP*.

Por último, un vistazo a los paquetes más usados, desvela donde está el cuello de botella de esta aplicación: es *H2*. La base de datos *H2* consume entre sus distintos paquetes más de un 48 % del tiempo de ejecución que hace que el recuperar datos de la base de datos sea muy costoso y que por lo tanto haya peticiones *HTTP* sin responder o que devuelven algún error.

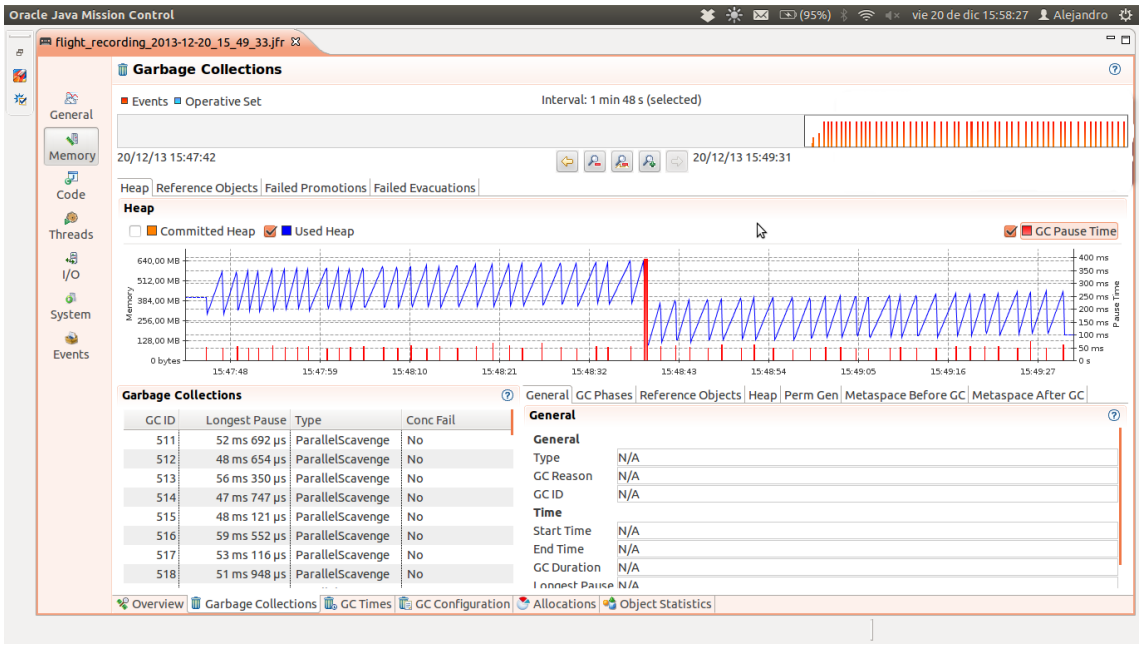

Figura 4: Gráfico de consumo de memoria.

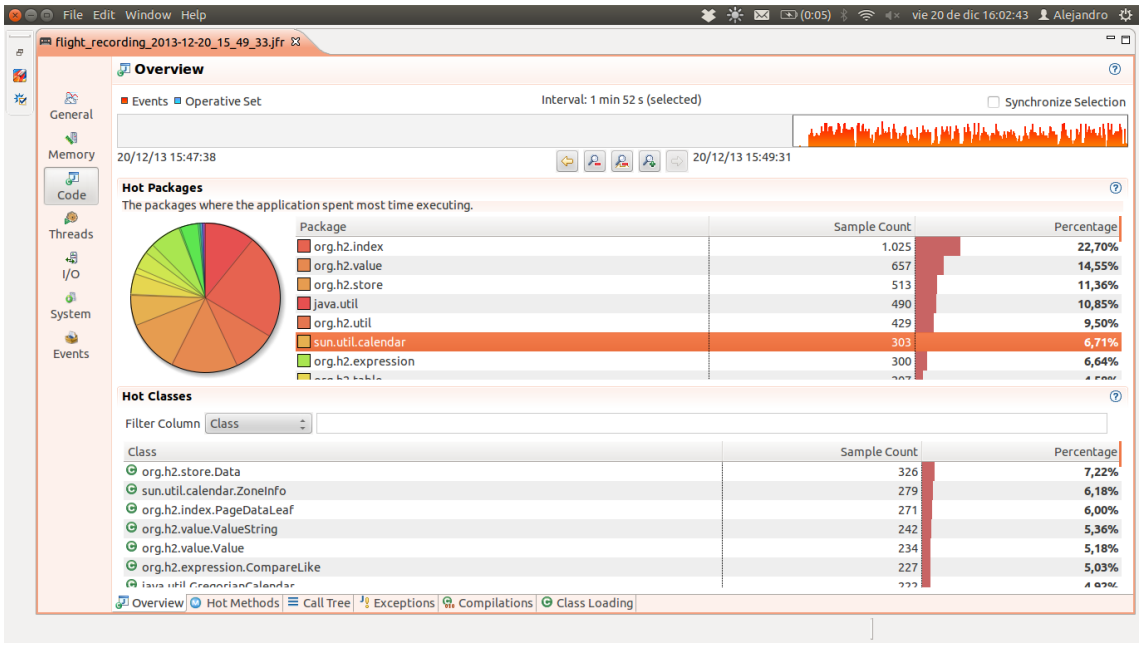

Figura 5: Gráfico de uso de paquetes.