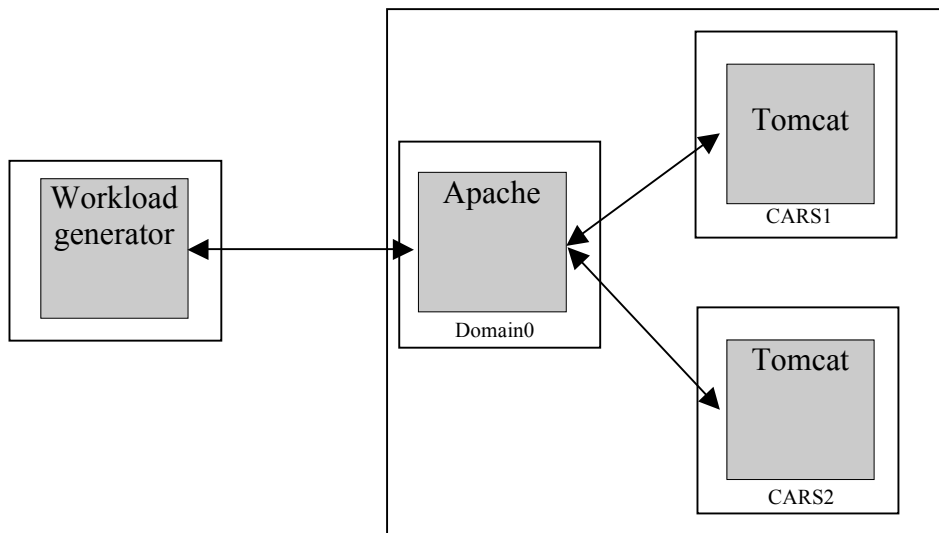


Configuració i Avaluació del Rendiment de Servidors (CARS)

Lab 8

Objective of the session:

Benchmark a dynamic web environment considering application-centric metrics:



where an Apache HTTP sever serves static contents, Tomcat runs dynamic web contents (JSP and Servlets). All the application data is stored in a backend DBMS (emulated in our lab). A workload generator emulates the load produced by a set of active web clients. Apache and Tomcat run in the same machine, while the workload generator resides in a different node.

Note: All the required files to complete this session can be found in <http://people.ac.upc.edu/dcarrera/cars/>

The required files are:
cars_client_sessio1_0809_final.tgz and
cars_server_sessio1_0809_final.tgz

Note: Most of the activities involved in this session take \$CARS_HOME as the root folder (corresponding to /home/alumne).

Part 0: Start the CARS virtual machines

As root, in the /root directory of the server machine run:

```
[$ROOT] xm list
```

To list the currently running VMs.

Now, change the allocated memory (300 Mb) for the Domain-0 in Xen.

```
[$ROOT] xm mem-set 0 300
```

It's time to start the first VM, that will boot with 2 virtual CPUs and 300MB of RAM.

```
[$ROOT] xm create CARS01.cfg vcpus=2 memory=300  
[$ROOT] xm create CARS02.cfg vcpus=2 memory=300
```

After that, you can check that the VM is running with

```
[$ROOT] xm list
```

Notice that if you look for the network interfaces in the domain-0 (using ifconfig) you'll find that for the domain X, there's a vifX.0 interface with an assigned IP. The command "route -n" will show the routes to access the VMs. In the case of CARS01, the address is 10.0.0.1. In the case of CARS02, it is 10.0.0.2.

You can log into the VM using an ssh session, with any of the following users/passwords:

```
alumne/sistememes  
root/cars
```

In the domain-0:

Uncompress the server files downloaded into \$CARS_HOME, using:

```
[$CARS_HOME] tar xvf cars_server_sessio1_0809_final.tgz
```

Copy the web application to the VM CARS0X:

```
[$CARS_HOME] scp -r app1 alumne@10.0.0.1:.  
[$CARS_HOME] scp -r app2 alumne@10.0.0.2:.
```

First: Setup the environment in the server running:

```
[$CARS_HOME] source env.sh
```

Part 1: Setting up httperf (workload generator)

Requirement:

- Httperf (httperf-0.9-cars.tgz)

1. Preparing the workload generator

Download the workload generator into the client machina:

```
[$CARS_HOME] tar xvzf cars_client_sessio1.0809Q1_final.tgz
```

A “httperf-0.9” folder is created. To be sure the environment works properly, configure and compile the application:

```
[$CARS_HOME/httperf-0.9] ./configure; make
```

The test can be run using:

```
[$CARS_HOME/httperf-0.9] ./starthttpperf.sh <load level>, for application 1
```

```
[$CARS_HOME/httperf-0.9] ./starthttpperf2.sh <load level>, for application 2
```

After executing this command, the workload generator will be running for 120s and will report, the data collected for the last 5s period.

Note: The load level represents the number of **new** sessions started per second.

Example: If the life-cycle time for each client is 100 sec. and 10 new seconds are generated per second, a maximum number of 1000 simultaneous clients will be observed, and 30 seconds after starting the test, 300 concurrent clients will be connected to the server.

Part 2: Testing

1. Set-up the server (Tomcat inside CARS VM, apache inside HOST)

Before running a test, make sure the servers (Apache and Tomcat) are configured as expected .

The parameters to configure for **Apache** are:

:

- **Thread pool size in** \$APACHE_HOME/conf/httpd.conf:
- Section “<IfModule **worker.c**>”

```
StartServers:           10
MaxClients:             <num threads>
MinSpareThreads:       <num threads>
MaxSpareThreads:       <num threads>
ThreadsPerChild:       <num threads / 10>
MaxRequestsPerChild:   0
```

- **Connection timeout in** \$APACHE_HOME/conf/httpd.conf :
- **Number of connections between Tomcat and Apache in** \$APACHE_HOME/conf/workers.properties:

```
worker.worker1.cachesize = <número de threads>
```

The parameters to configure for **Tomcat** are:

- **Thread pool size in** \$TOMCAT_HOME/conf/server.xml (AJP 1.3 connector)

```
maxThreads = <#threads>
minSpareThreads = <#threads>
maxSpareThreads = <#threads>
```

To start the servers, you must run:

\$APACHE_HOME/bin/apachectl start | stop, to start and stop Apache

\$TOMCAT_HOME/bin/catalina.sh run, to start Tomcat (it is stop pressing Ctl+C)

2. Run the experiments:

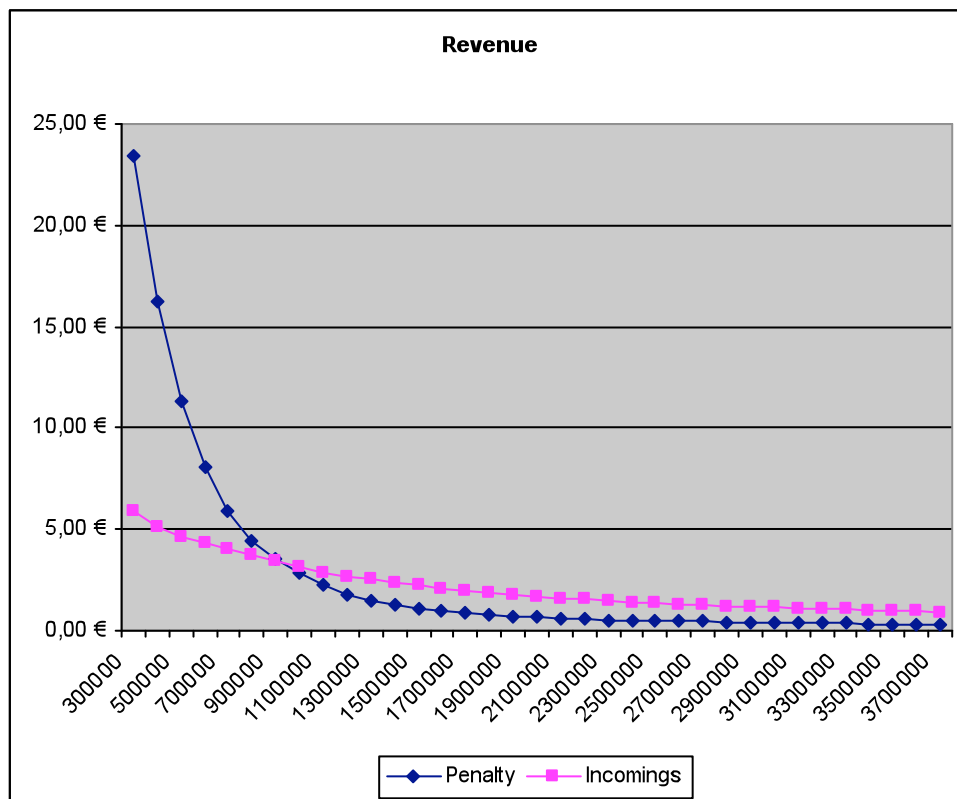
EXPERIMENT 1: Determining the profit of a system

Imagine you're the system administrator of a large computing facility composed of a number of nodes. One of your customers is about to sign an agreement with you to deploy his application on your system. At this point you have to decide the SLA that you will accept to sign, as well as the amount of money to get if the SLA is met and the money to pay if the SLA is missed.

For this purpose, you can use a spreadsheet to be provided in the lab. On it, you'll find some cells representing your incomings and your penalties, amongst others. The input you have to provide is the logfile resulting from running httperf at a certain load level.

After that, the spreadsheet will calculate how many of the request have been processed within the expected response time, how many have missed the SLA and from these, how many have been processed within 110%, 120%, 130%, 140%, 150%, 160%, 170%, 180%, 190% and 200%+ of the response time defined in the SLA.

This evaluation is performed for several possible SLAs, and an example of the data that the spreadsheet will calculate for you can be seen below:



The spreadsheet is configured to calculate:

- Money you get for every 1000 requests processed within the SLA:
 $X \text{ €} * (\text{baseline time} * 2 / \text{SLA})$
- Penalties:
 - o 1000 requests 110% of the response time: 1€
 - o 1000 requests 120% of the response time: 1€
 - o 1000 requests 130% of the response time: 2€
 - o 1000 requests 140% of the response time: 2€
 - o 1000 requests 150% of the response time: 2€
 - o 1000 requests 160% of the response time: 3€
 - o 1000 requests 170% of the response time: 3€
 - o 1000 requests 180% of the response time: 4€
 - o 1000 requests 190% of the response time: 4€
 - o 1000 requests 200+% of the response time: 5€

Your requested to:

- a) Run the following configurations and load levels, collecting the apache logs after each execution as well as the output of the workload generator. The logs must be later used to feed the spreadsheet.
 - 1) [Apache pool size=20, Apache-Tomcat connections=10, Tomcat pool size=10, Keep-alive timeout=15, httpperf load=5]
 - 2) [Apache pool size=120, Apache-Tomcat connections=200, Tomcat pool size=200, Keep-alive timeout=15, httpperf load=5]
 - 3) [Apache pool size=280, Apache-Tomcat connections=350, Tomcat pool size=350, Keep-alive timeout=15, httpperf load=12]
 - 4) [Apache pool size=300, Apache-Tomcat connections=400, Tomcat pool size=400, Keep-alive timeout=15, httpperf load=13]
 - 5) [Apache pool size=300, Apache-Tomcat connections=400, Tomcat pool size=400, Keep-alive timeout=15, httpperf load=14]
- b) Decide the minimum SLA you would accept to sign in each case, assuming that the proposed configuration is the best you expect to be able to offer to your customer.

EXPERIMENT 2: Determining the profit of a system, running 2 Tomcat instances

Repeat the experiment 1 but running 2 applications instead of 1. Decide the sample points and find the maximum load and the SLA to sign for each application to maximize the overall profit for your company. Notice that you should allocate resources to your VMs correspondingly to your target loads.

Note: Remember to put the application logs into two different CSV files.

EXPERIMENT 3: Utility functions

For application 1, tune the system to run the test with load 10, 11, 12, 13, 14 and 15 properly. After that, fill a spreadsheet with the average response time observed and draw the following utility function, for SLA levels 100, 150, 200, 250, 300 and 500 ms.

$f(\text{actual response time}) = (\text{actual response time} - \text{response time goal}) / \text{response time goal}$