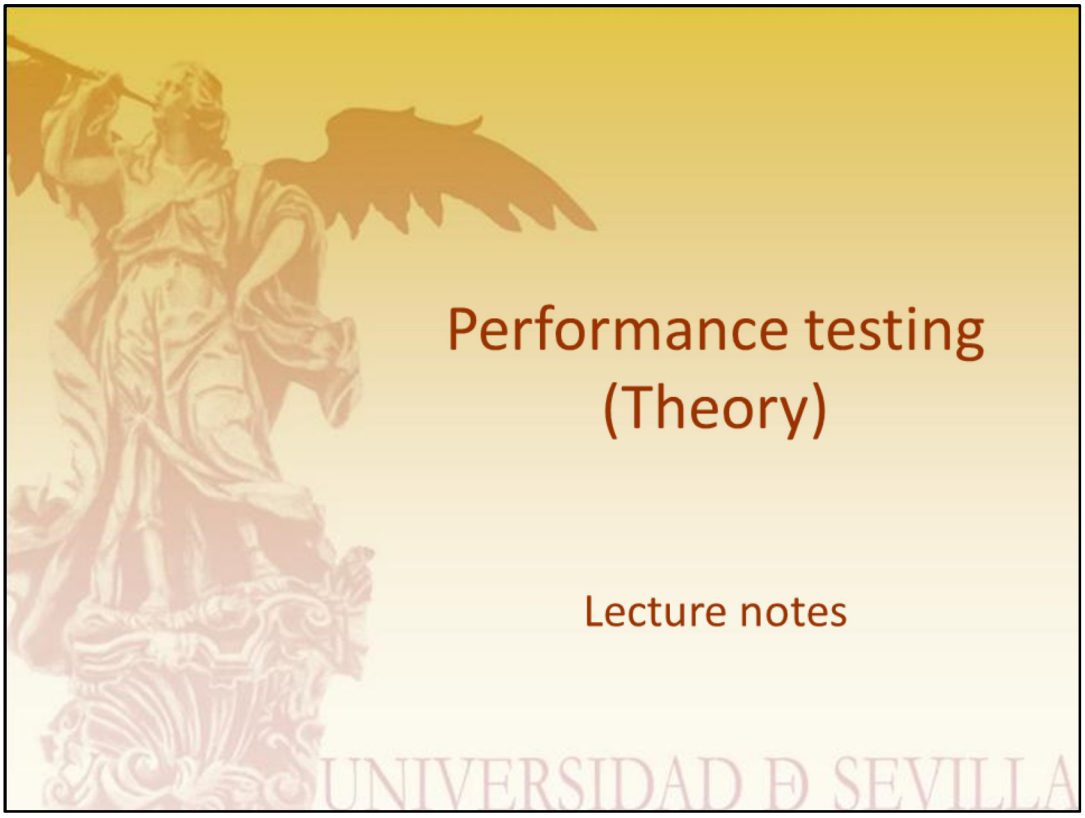


Welcome to this lecture! Today, we're going to work on performance testing.

--

What's performance testing about?

UNIVERSIDAD Ð SEVILLA

2

As usual, we start with a question: what's performance testing about?

This is a good definition

Performance testing refers to tests to determine how a system performs in terms of stability and responsiveness under a particular workload

UNIVERSIDAD Ð SEVILLA

3

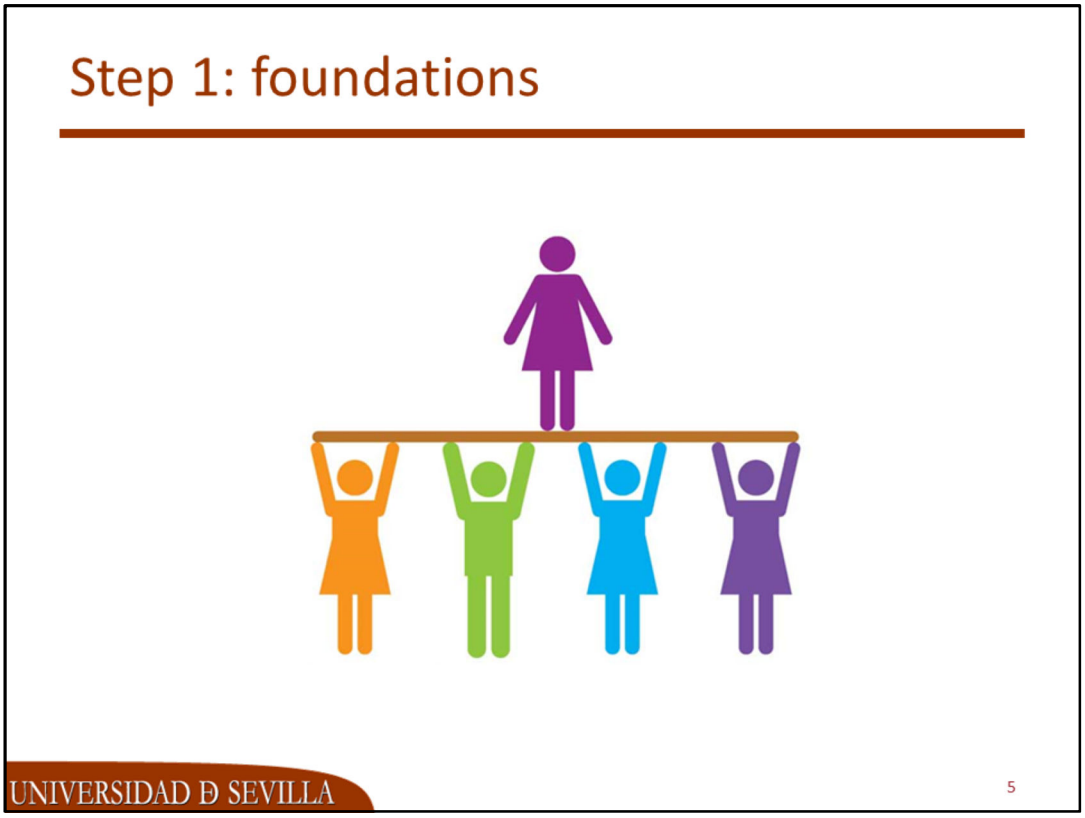
Performance testing, aka benchmarking, refers to a series of tests that are conducted to determine how a system performs in terms of stability and responsiveness when it's confronted with a particular workload. By stability we mean the ability of the system to perform without any failures; by responsiveness we mean its ability to respond to requests in a reasonable time.

What do I have to know?

UNIVERSIDAD Ð SEVILLA

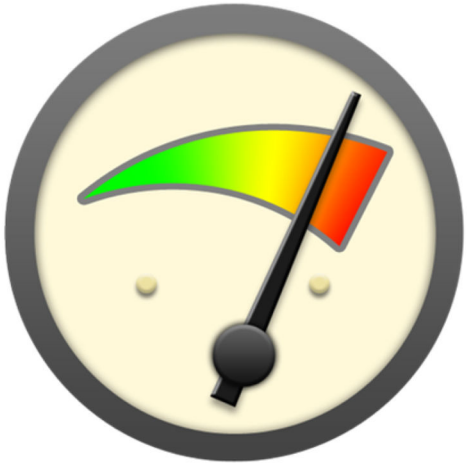
4

Let’s now report on what you have to know regarding performance testing.



First, and foremost important, you need to know a few foundations about performance testing.

Step 2: the testing tool



UNIVERSIDAD Ð SEVILLA

6

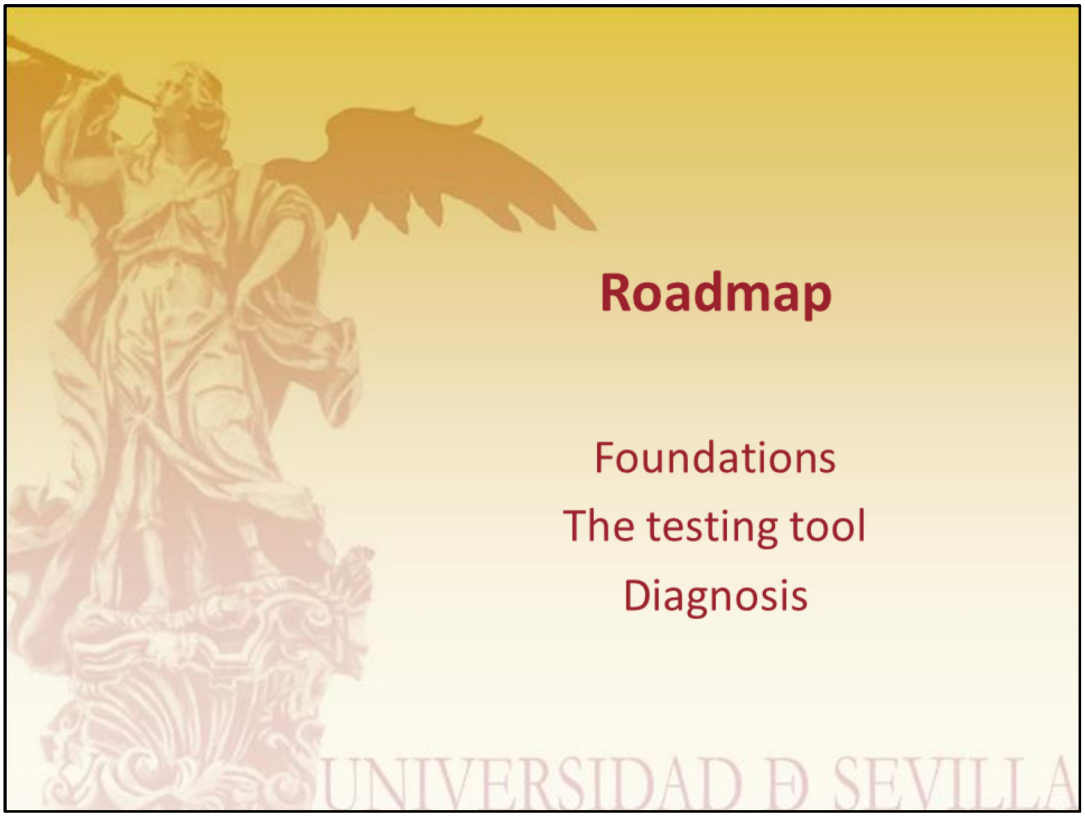
Then you need to know a testing tool that allows you to simulate a workload and measure how your system performs.

Step 3: diagnosis

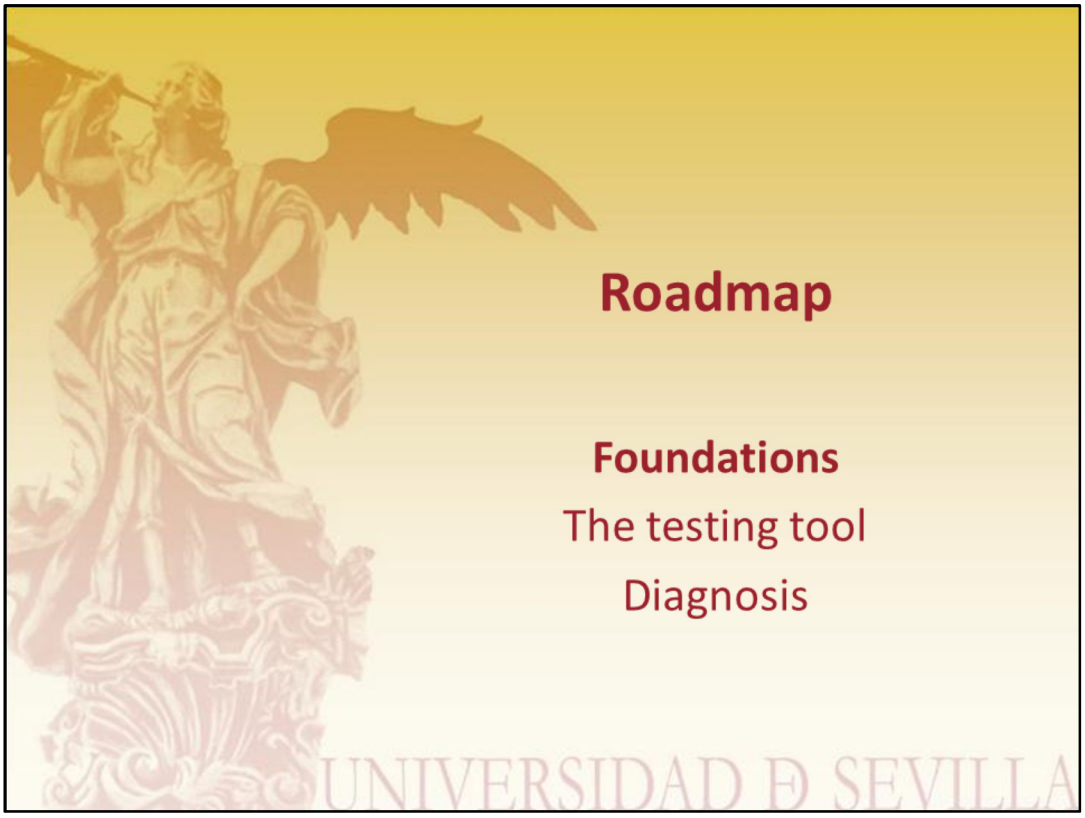
UNIVERSIDAD Ð SEVILLA

7

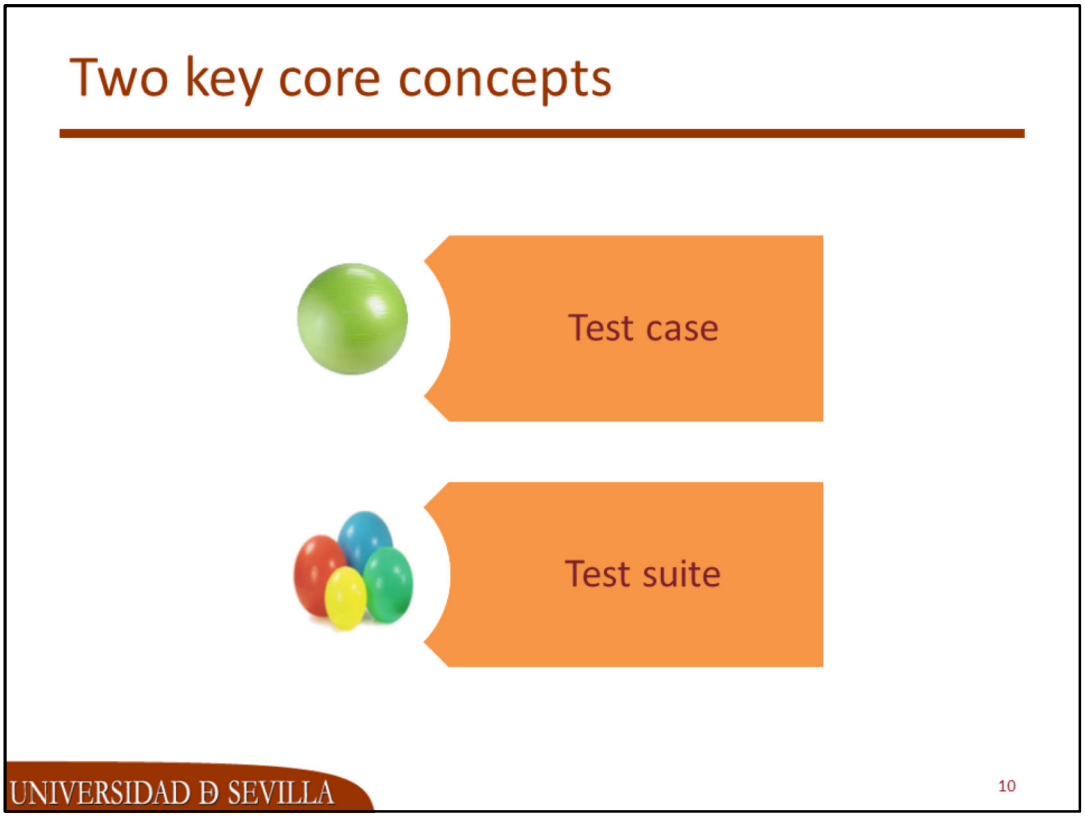
Finally, you need to know how to diagnose your system when it performs unstably or poorly.



It shouldn't then be surprising to you that this is our roadmap for today's lecture. We'll first provide a foundation, then we'll introduce the testing tool that we're going to use, and, finally, we'll report a little on diagnosis.



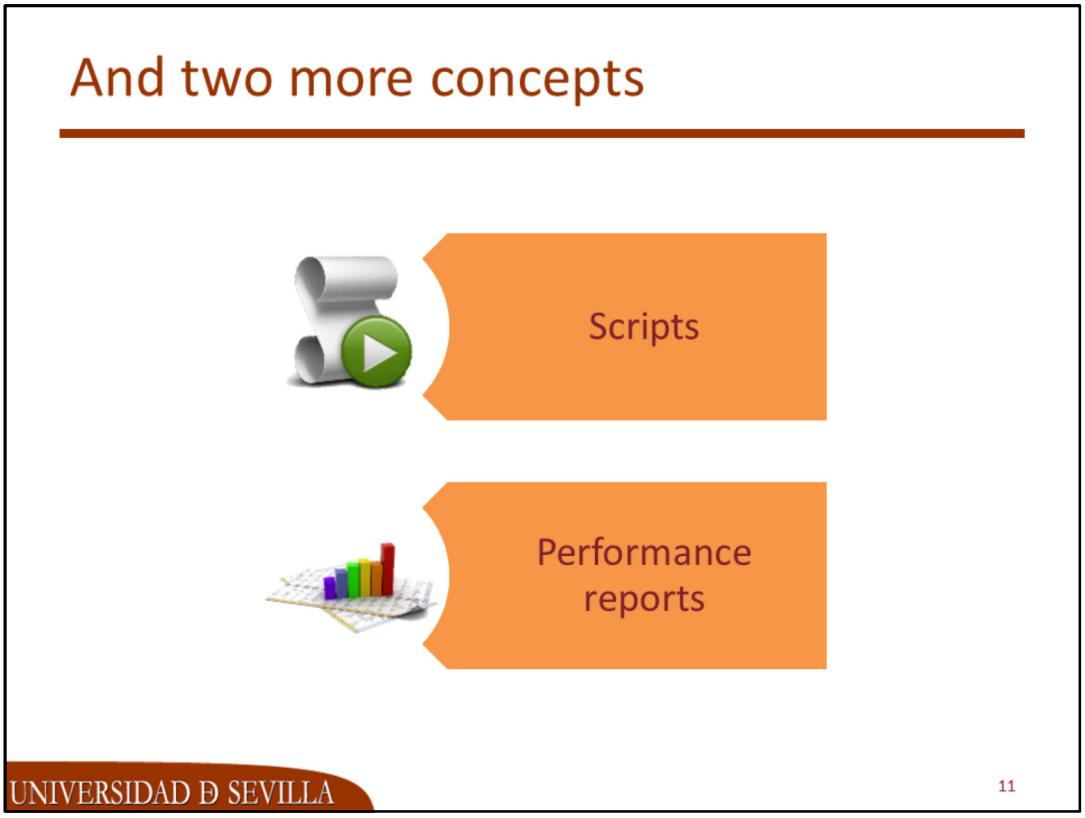
Let’s start with a few fundamental concepts, not many, we promise.



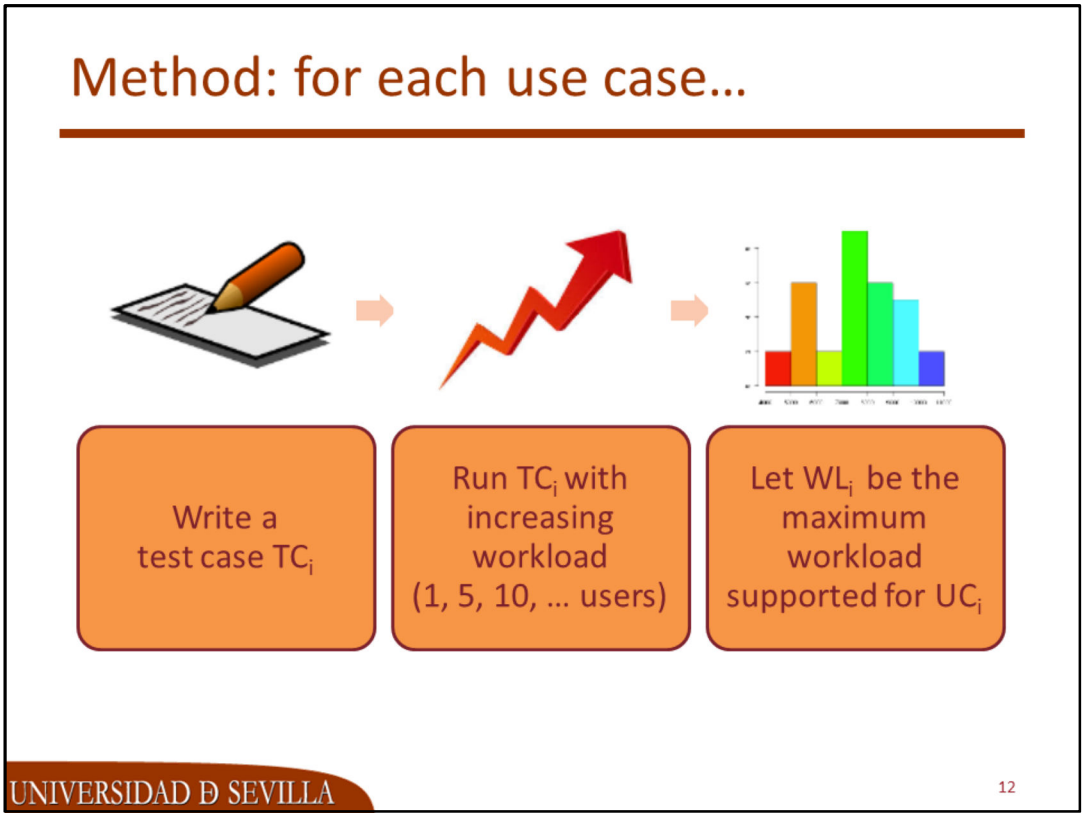
This slide introduces a couple of key core concepts, namely:

- Test case: it's a test that emulates a user performing a use case.
- Test suite: it's a collection of test cases.

Note that these core concepts have been borrowed from the lesson on functional testing since they also apply to performance tests. What differentiates functional test cases or suites and performance test cases or suites is their goal and how they are implemented. Please, keep reading.



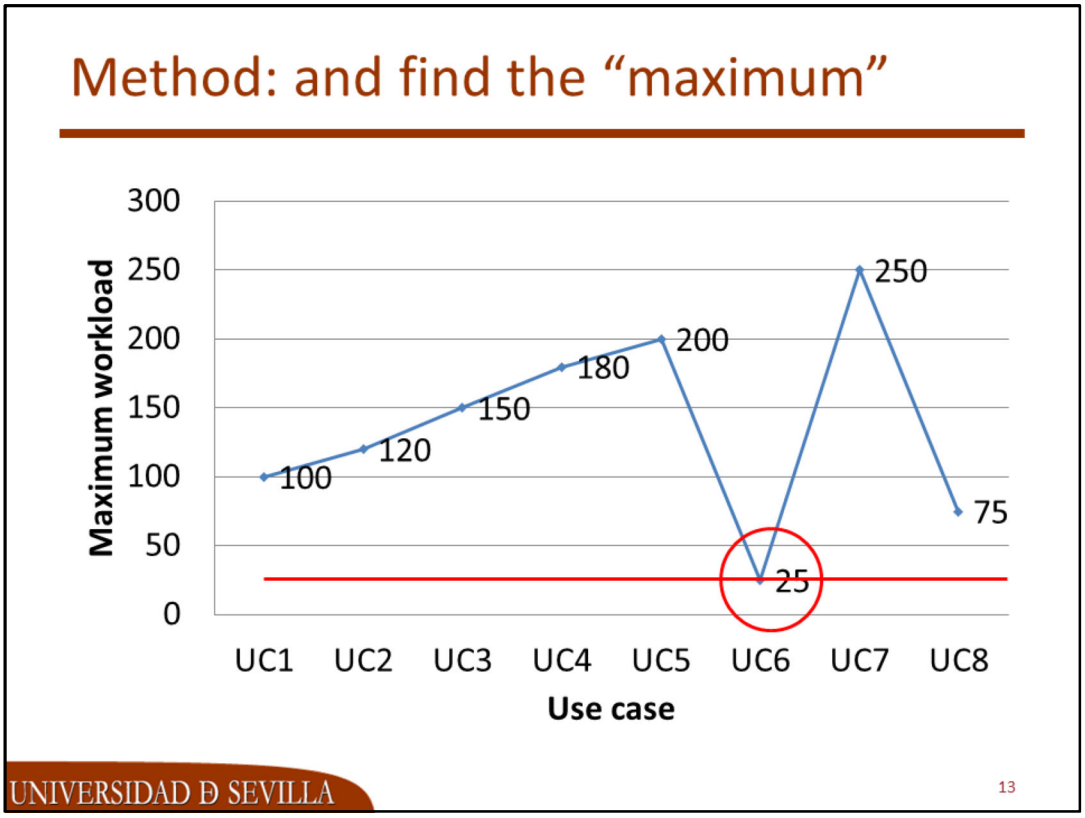
Internally, test cases are implemented using scripts that emulate the interactions that a user performs with a system when he or she's executing a use case. Test cases also have so-called performance reports, which basically measure the throughput of the system in terms of use cases completed per unit of time.



Another important foundation is our method to implement performance tests. Basically, you have to perform the following steps for every use case:

- Write a test case. We're going to provide a guideline in the following section. Simply assume that they are easy to write.
- Run it repeatedly by increasing the workload gradually. Increasing the workload means that you simulate a monotonically increasing number of users who are using your system. One, five, ten users; that works pretty well. If your system can handle that workload gracefully, then try larger increments… but don't start with 1000 users since this will very likely take your system down.
- Jot the maximum workload down. The maximum workload is the maximum number of concurrent users that your system supports regarding a particular use case while it keeps responsive and doesn't crash.

As a result, you'll get a chart in which you document the maximum workload that your system can handle on a per-use-case basis.



And then compute the maximum workload that your system can handle, which paradoxically isn’t max {WL_1, WL_2, … WL_n}, but min {WL_1, WL_2, … WL_n}. Note that the purpose of performance testing is to provide a maximum workload that you can guarantee that your system can handle gracefully. In the system in the figure, you can guarantee a maximum workload of 25 concurrent users. Ok, your system supports up to 250 concurrent users executing use case UC7… but the problem is that you cannot guarantee that your system can handle 250 concurrent users because if more than 25 are executing use case UC6 concurrently, then your system becomes unresponsive or simply crashes. So the maximum workload that your system can guarantee is 25 concurrent users.

Recall our configurations?

Developer's configuration

(Functional testing goes here)

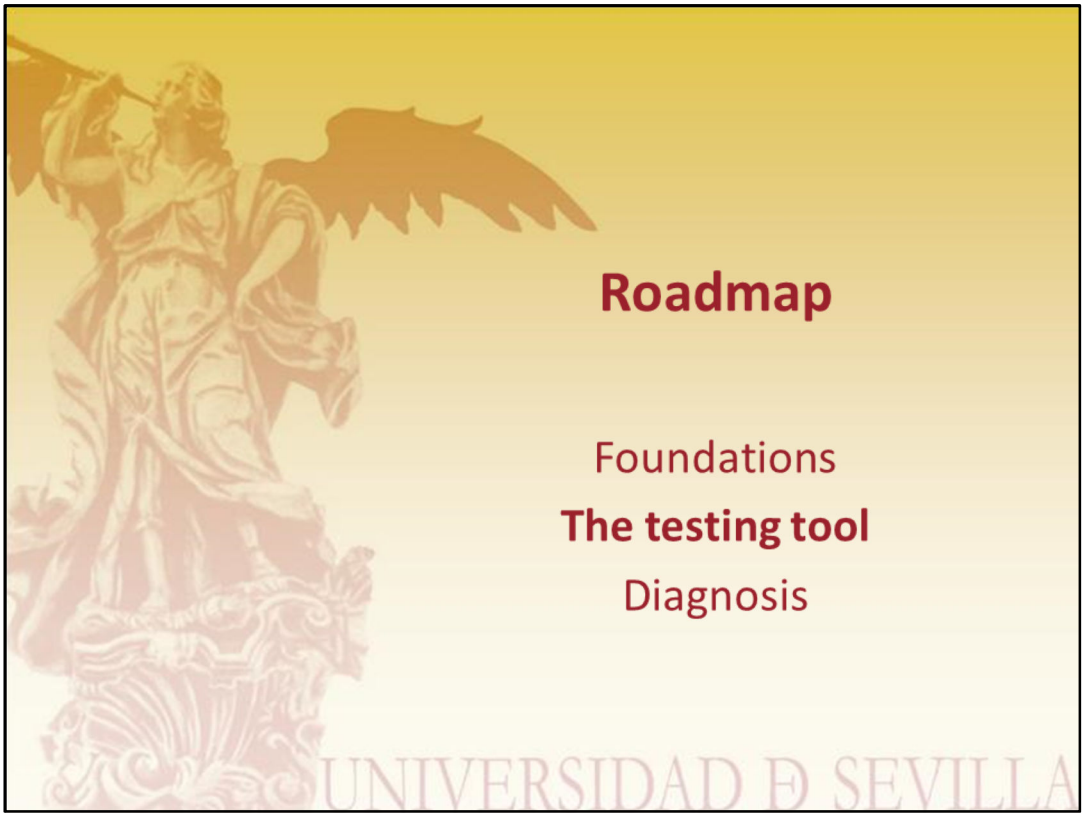
Pre-production configuration

(Performance testing goes here)

UNIVERSIDAD Ð SEVILLA

14

Last, but not least, please, recall that we have a developer's configuration and a pre-production configuration. The former provides every tool you need to develop your web information systems and the latter's a clone of your customer's server. You can use the former to conduct your functional tests and the latter to conduct your performance tests.



Let’s now report on the performance testing tool that we’re going to use in this subject.

Nice to meet you!

UNIVERSIDAD D SEVILLA

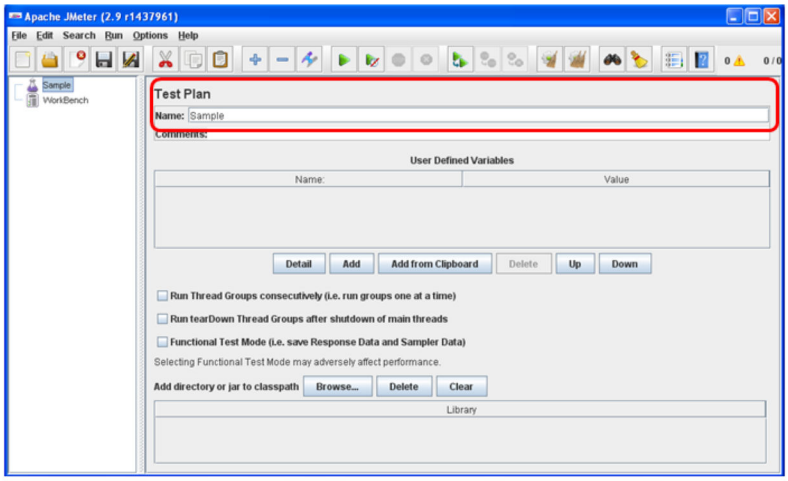
16

We're going to use Apache jMeter, which is quite a simple tool; but it's very popular because it helps create performance tests and run them very easily.

This is the roadmap that we're going to explore regarding jMeter: first, we'll explore how to create a test case, then how to record a script, how to tidy it up, how to create performance reports, and, finally, we'll explore how to run your performance tests.

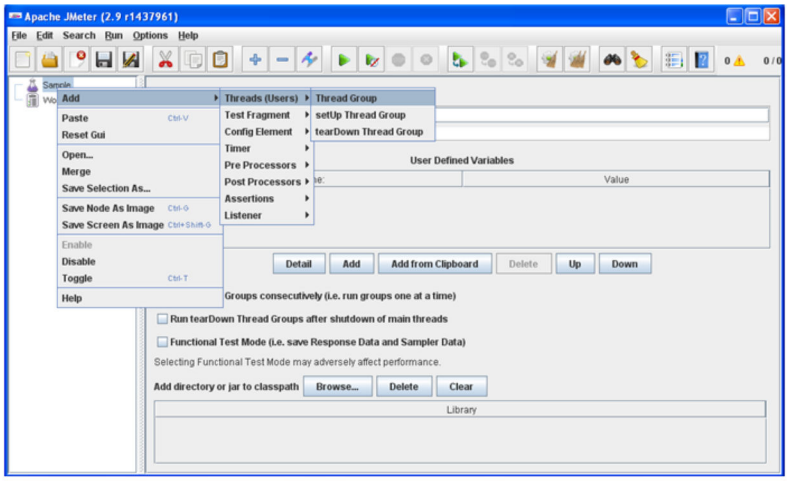
Let’s start with how to create a test case.

Start jMeter up



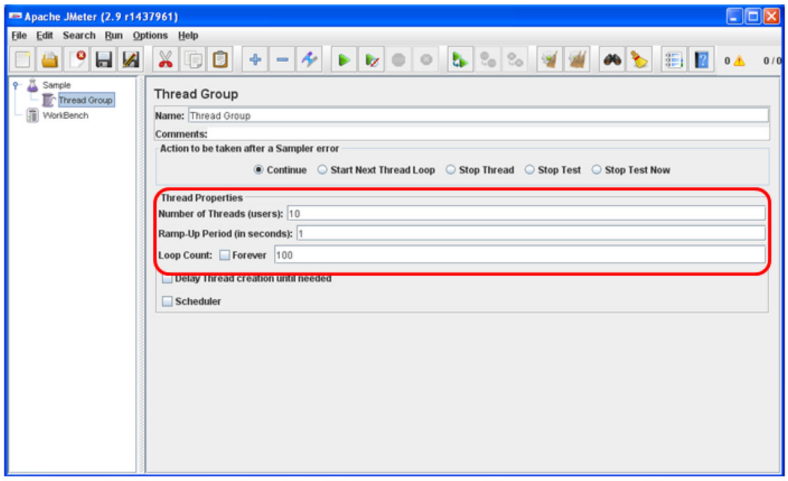
This is how jMeter looks like. By default, it starts up with a blank performance test called “Test Plan”. Change the name to an appropriate identifier, e.g., “Sample”. You don’t have to configure anything else here.

Create a thread group



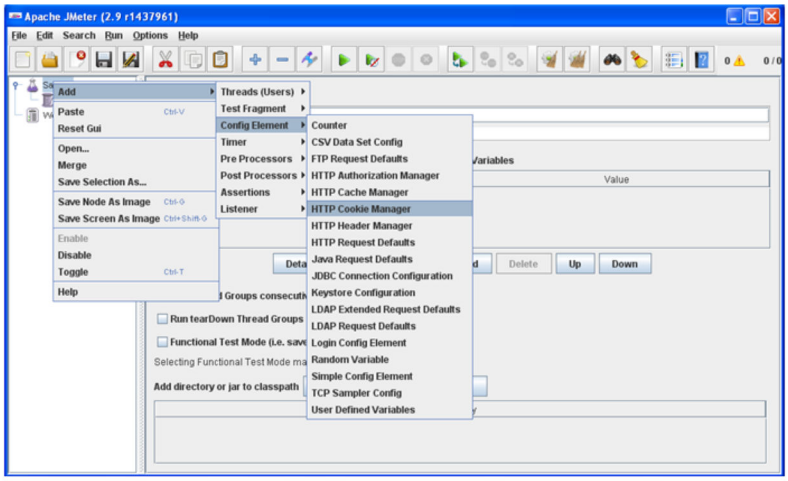
After changing the name, we need to create a so-called thread group. Simply put: a thread group will help us simulate a number of concurrent users who are executing typical use cases on our system.

Configure your thread group



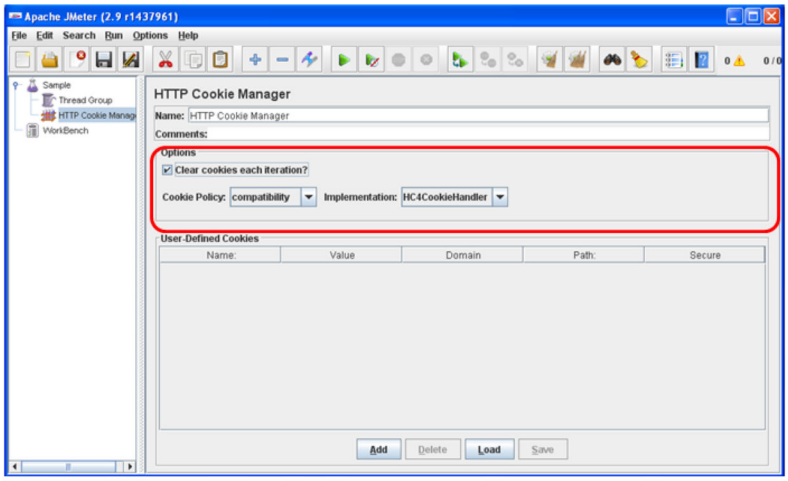
The minimum configuration that you must perform is to change the number of threads, which indicates the number of users that will be simulated, and the loop count, which indicates how many times each thread will execute the use case with which we're going to test our system (we'll provide additional details on this later). In our example, we set these parameters to 10 and 100, respectively; that means that jMeter will simulate 10 users who execute a use case 100 times each. There's a parameter called ramp-up period that is sometimes useful. Formally speaking, if there are *T* threads and the ramp-up period is *R* seconds, then jMeter will take *R* seconds to get the *T* threads up and running; each thread will start *R* / *T* seconds after the previous thread was started. For instance, if there are 30 threads and the ramp-up period is 120 seconds, then each successive thread will be delayed by 120 / 30 = 4 seconds.

Add an HTTP cookie manager



We also need to add an HTTP cookie manager to our benchmark. Please, recall that our application uses cookies to, e.g., persist the language in which the interface must be rendered or Tomcat's session number. Thus, we need an HTTP cookie manager in jMeter to store them.

Configure the HTTP cookie manager



There's little to configure regarding the HTTP cookie manager: check that you wish the cookies to be cleared on each iteration, set the cookie policy to "compatibility", and set the implementation to "HC4CookieHandler", which works pretty well.

Creating a test case, wasn't difficult at all, was it? Now we need to explore how to record a script.

What's a script?

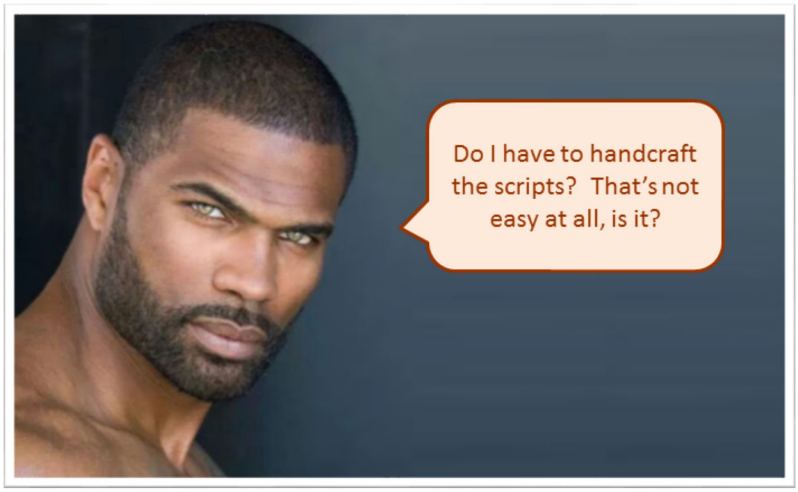
A script's a sequence of interactions that a user performs when he or she's executing a use case

UNIVERSIDAD Đ SEVILLA

25

A script is a sequence of interactions that a user performs when he or she's executing a use case. Simply put, a list of HTTP requests that he or she makes to the server. We need to add scripts to our test cases so that jMeter can reproduce them and simulate a number of users who are interacting with our system.

This is a good question



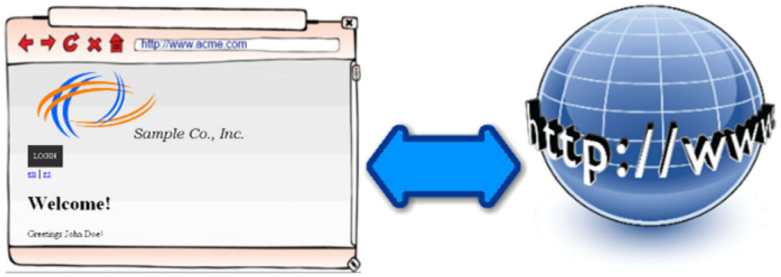
UNIVERSIDAD D SEVILLA

26

Handcrafting such scripts is not easy at all. More than that: it's very error prone.

Fortunately, jMeter can record them automatically while you interact with your systems. jMeter uses proxies to record your scripts, so we'd better learn a little theory about them.

Direct connections



UNIVERSIDAD Ð SEVILLA

28

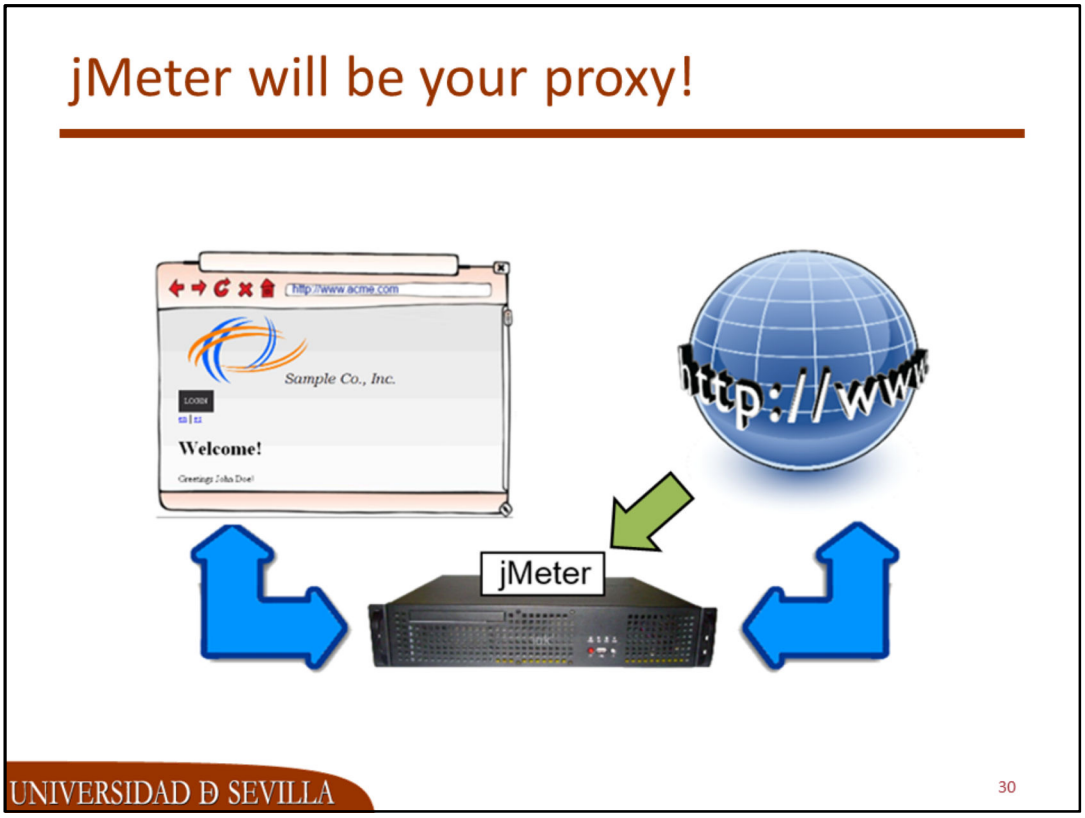
Typically, when you make your browser for an address like “http://www.acme.com”, it sets up a network connection to the corresponding server and sends an HTTP GET request through it; the server then produces an answer, returns it, and the browser renders it on the screen. Pretty simple. That’s what experts call a direct connection. We explored this process in one of our earliest lectures.

Proxied connections

UNIVERSIDAD Ð SEVILLA

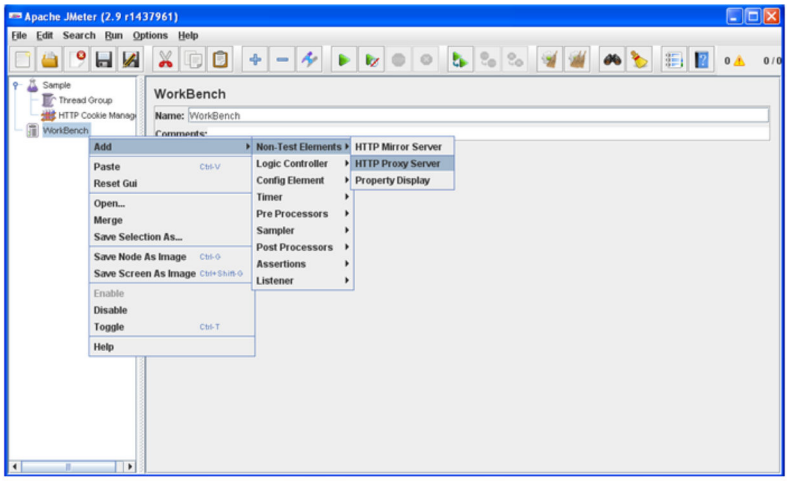
29

But there are also so-called proxied connections. Such a connection goes through a proxy, which is a device that acts as an intermediary between your browser and the actual site to which you're sending a request. (Note that a proxy can be a physical machine or a software process.) Having a proxy has a number of advantages; for instance, a proxy allows your machine to be in a private network to which other machines in the world do not have access, which helps your organisation be more secure; a proxy may also cache common requests that dozens of people make daily without hitting the actual server, which decreases the bandwidth that you consume. These are the reasons why proxies are very common in professional scenarios.



jMeter will be your proxy! The idea is that when you make your browser for "http://www.acme.com", the request will go through jMeter, which will then be able to easily record your interactions with the actual site.

Add an HTTP proxy

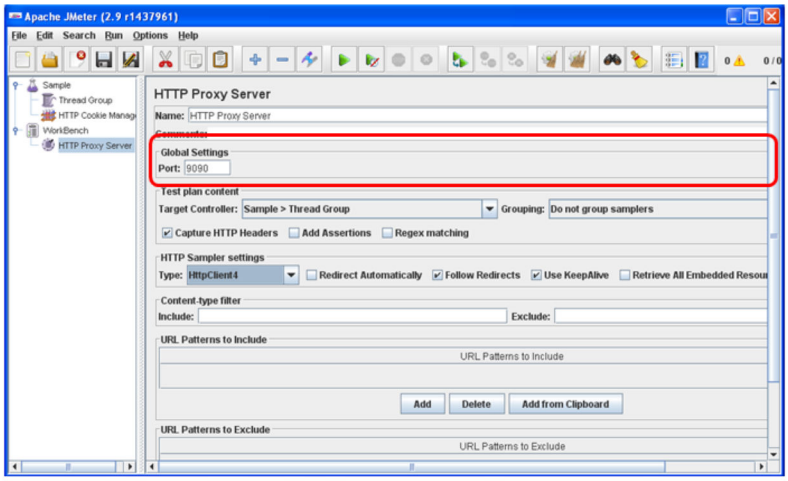


UNIVERSIDAD D SEVILLA

31

To convert jMeter into a proxy, you have to right click on “Workbench” and add an HTTP proxy server.

Configure the HTTP proxy

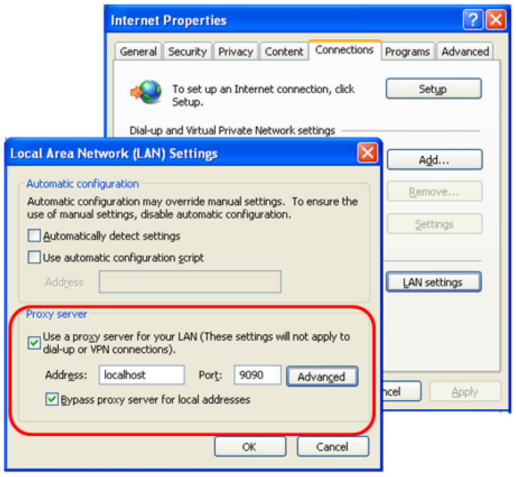


UNIVERSIDAD D SEVILLA

32

You only need to change the port to 9090 in order to avoid confusion with the HTTP port that we use in development configurations. That's all you need to configure.

Change your system's proxy



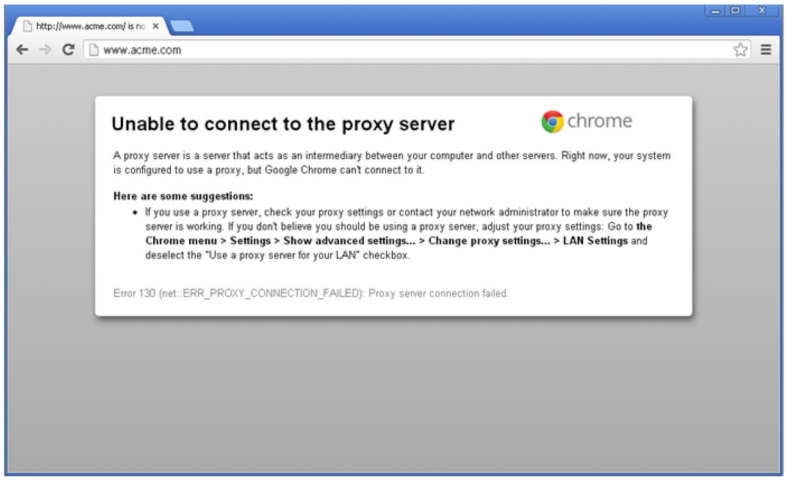
UNIVERSIDAD D SEVILLA

33

The next step is to change your operating system's proxy. To do so, you must open your local area network settings by clicking on Start > Control Panel > Network and Internet Connections > Internet Options > Connections > LAN settings. You'll get a dialog like the one in this slide, where you must check that your proxy server is at address "localhost" and port "9090". Please, don't forget to check that you wish to bypass the proxy server for local addresses; that means that if you make your browser for "http://www.acme.com", the connection will be proxied, but if you make it for "http://localhost" it'll result in a direct connection. The reason why local addresses must be bypassed will become clear in a few slides.

NOTE: please, consult your operating system's documentation to find out how proxies are configured. Unfortunately, there's not a standard way to configure your operating system's proxy. That's why we can't provide you with a guideline.

Check that … nothing works



Let's try the proxy. Make your browser for "http://www.acme.com". You should get this error message, which indicates that your browser's attempting to establish a proxied connection to the site, but the proxy's not responding. That shouldn't be surprising: before, we configured the proxy, but we didn't start it.

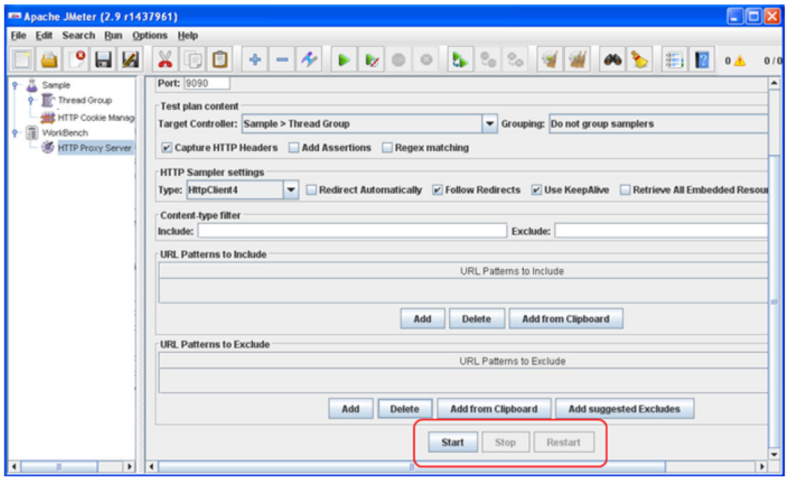
Except for localhost requests

UNIVERSIDAD D SEVILLA

35

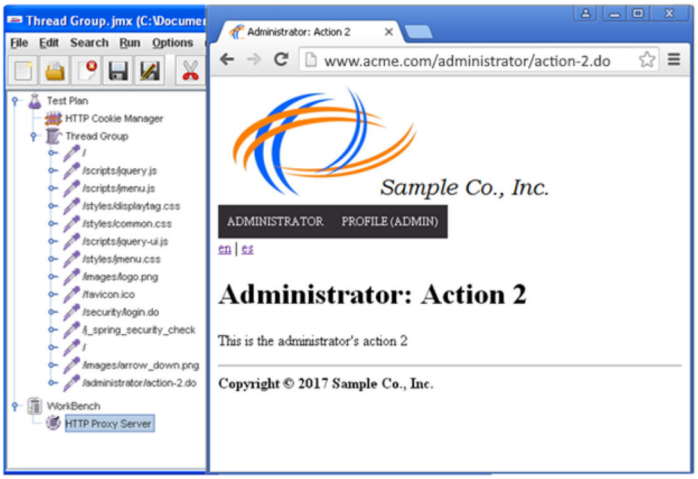
Before starting the proxy, please, make your browser for “http://localhost”. That should work because this is a local address and you’ve instructed your operating system not to proxy local addresses.

Let’s start jMeter’s proxy up



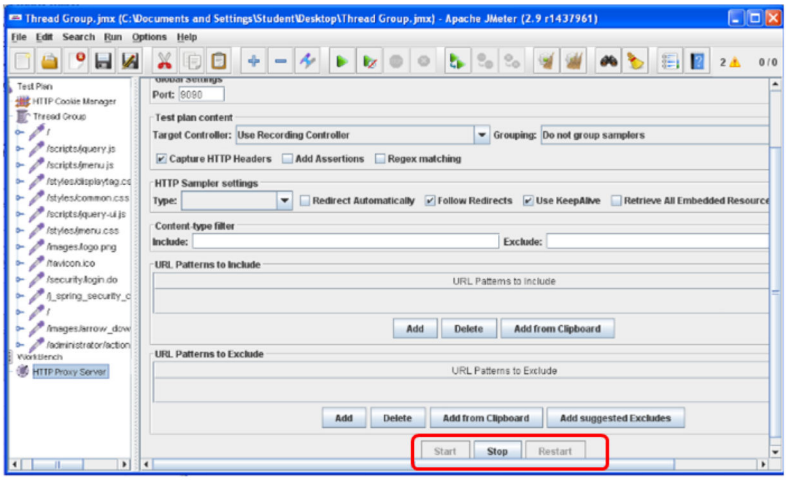
Ok, it’s now time to start jMeter’s proxy. Get back to jMeter, scroll a little down, and press the “Start” button. Apparently, nothing happens, but get back to your browser.

Let's record a simple script



Make your browser for "http://www.acme.com", log in to the system, and execute a few options from the main menu. Can you see that jMeter's recording your interactions? Every time you make a request to the server, jMeter intercepts it and records it.

Stop the proxy



UNIVERSIDAD D SEVILLA

38

Ok, play for a little and record a small script. Press the “Stop” button when you’re done.

Let’s now talk a little about tiding your scripts up.

Tiding your scripts up

UNIVERSIDAD Ð SEVILLA

40

Unfortunately, the scripts that you record are not usually ready to be executed. Before that, you need to remove spurious entries and add sensible human delays.

Let’s talk about spurious entries

UNIVERSIDAD Ð SEVILLA

41

Let’s first talk a little about spurious entries.

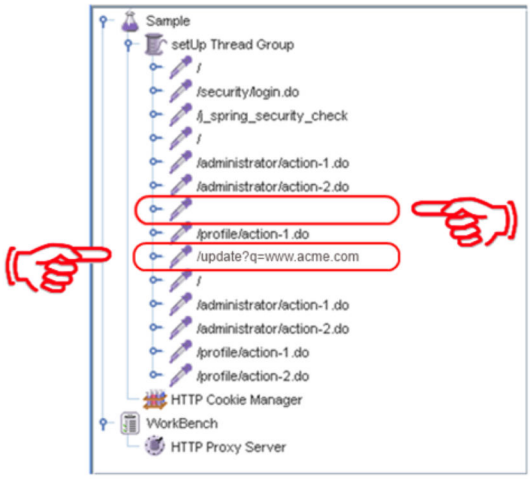
This happens very often!

UNIVERSIDAD Ð SEVILLA

42

Please, review your script carefully when you finish recording it. You must check that every entry in your script corresponds to your web information system; that is, you must check that there aren't any spurious entries.

These are spurious entries

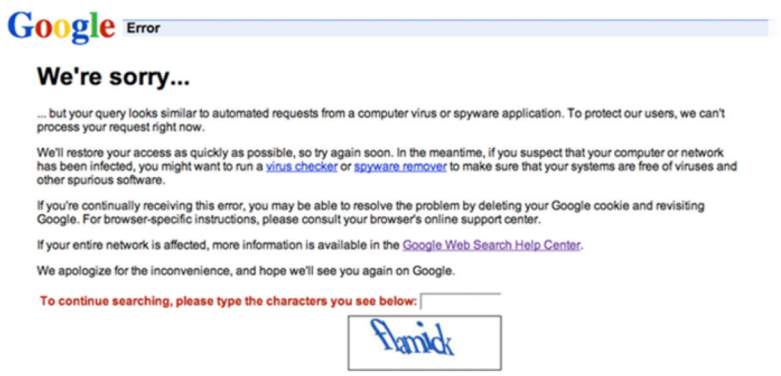


UNIVERSIDAD D SEVILLA

43

Spurious entries are entries like the ones that we have highlighted in this picture. Note that there is an entry without a path and an additional entry that corresponds to a path that doesn't exist in our system. These entries typically correspond to background requests that your browser performs. In this case, we were using Chrome and the spurious requests were sent to Google's servers. Do you know what happens if you simulate a workload of 10 users running the script in this slide 100 times each? That means that you're performing at least 2,000 requests to Google's servers in a very short period of time.

Be careful with such entries



UNIVERSIDAD Ð SEVILLA

44

And this is how Google's servers protect themselves from your script: Google will kick you out! The first few times, Google allows you to recover the service by typing in a captcha; but if you persist, Google will definitely ban your IP address from having access to their services. So you'd better review your script and remove entries that involve other companies' servers.

It's not *your* problem, it's *our* problem

UNIVERSIDAD Đ SEVILLA

45

Please, note that the IP addresses in our laboratories are limited. Before running a script, you must make sure that it does not include any calls to any companies' servers or otherwise you might ban a University IP address and prevent other students from using these companies' services. **Please, do check your scripts twice or three times before running them!**

Let’s talk about adding delays

Let’s now talk about adding sensible human delays.

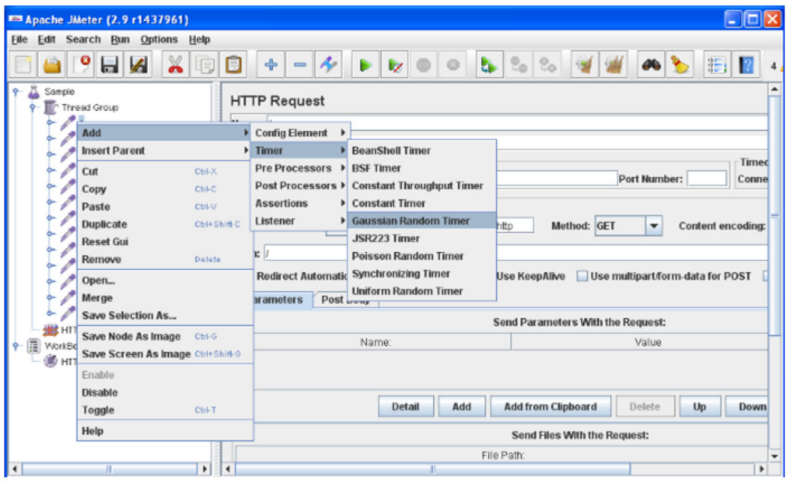
The scripts run at full speed!

UNIVERSIDAD Ð SEVILLA

47

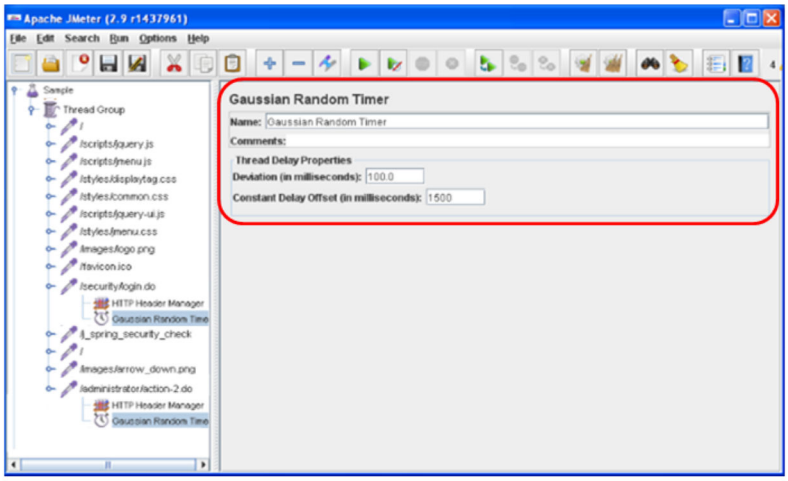
If you execute a script that has just been recorded, you'll easily find that it runs at full speed. Unfortunately, the proxy records the interactions with your web information system, but not the delays.

Add timers to delay your scripts



In order to introduce a delay in your scripts, you need to add timers to your entries. Click on the entries that represent human interactions and select “Add > Timer”. There are a variety of timers available, but we recommend that you should use Gaussian timers.

Configure your timers

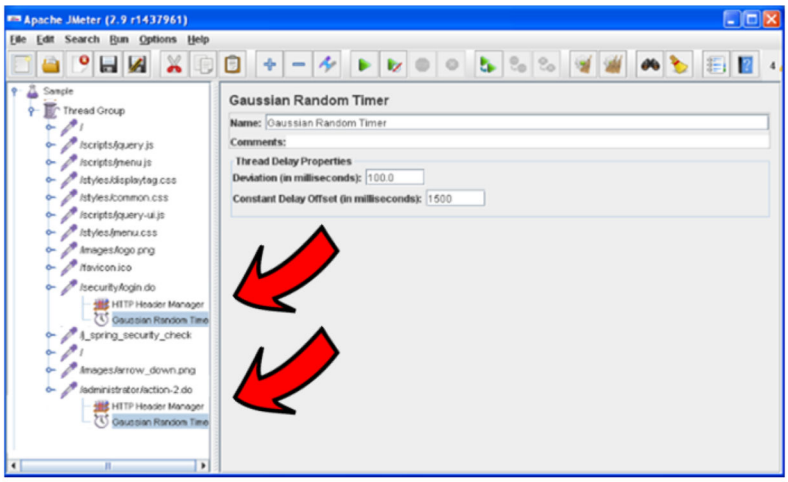


UNIVERSIDAD D SEVILLA

49

Configuring a Gaussian timer's very easy: you just need to provide a deviation and an offset. The terminology is a little odd, but it isn't that difficult to understand: a Gaussian timer with deviation *a* and offset *b*, delays the execution of a script for a period of time that's distributed according to a normal distribution with mean *b* milliseconds and deviation *a* milliseconds. We recommend that you should set the deviation to 100 milliseconds and the offset to 1500 milliseconds; these figures are somewhat realistic and they typically work quite well.

Add them to human interactions only!



UNIVERSIDAD D SEVILLA

50

Note that you have to delay only in the entries that are related to user interactions. For instance, after a user requests a page, he or she typically waits for a little until he or she interacts again. You don't have to introduce a delay when the interaction is performed automatically by the browser, e.g., to retrieve scripts, CSS files, or other resources because those interactions are actually performed at full speed.

It’s now time to learn how to add performance reports to your test cases.

Performance reports

UNIVERSIDAD Ð SEVILLA

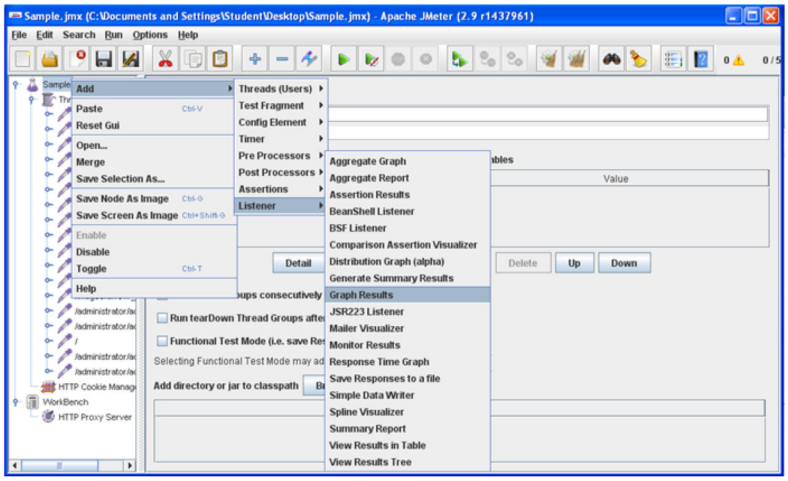
52

A performance report's a report that provides a variety of performance measures that are computed while your test cases run. jMeter provides a variety of reports, but we're going to focus on two of them exclusively, namely: graph reports and aggregate reports.

Performance reports

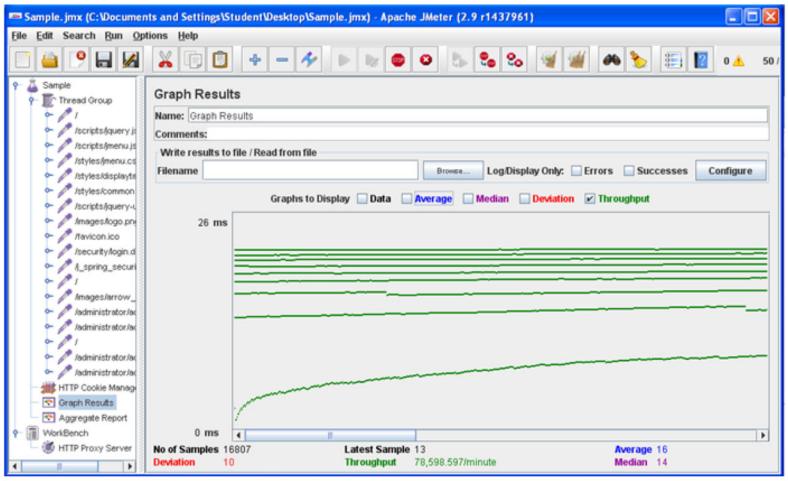
Let’s first talk about graph reports.

Add a graph report



Reports are introduced by means of so-called listeners. A listener's an object that listens to the execution of your test cases, collects measures, and creates reports. Please, add a "Graph results" listener to your test case so that you can have a graph report.

A sample graph report

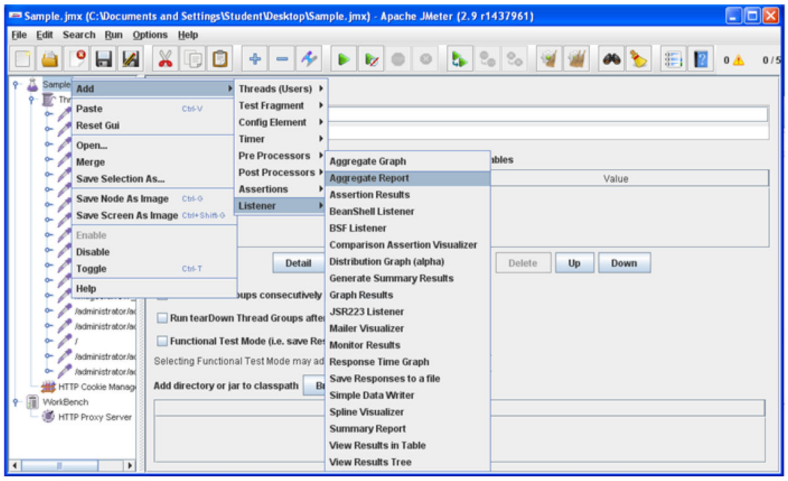


When you run your test case, you'll get something similar to the screenshot in this slide when you click on your graph report. The green line shows the throughput, that is, the average number of times per unit of time that jMeter is able to execute the script that you've recorded. In this chart, the throughput is roughly 78,500 executions per minute. (Wow, that's quite an impressive figure!) Each line represents an iteration; the line at the bottom clearly shows that the initial throughput is small because the threads are starting, and it increases as the workload increases and your system caches start working. You can also show the average time to run the script (in blue), the median time (in magenta), and the deviation time (in red).

Performance reports

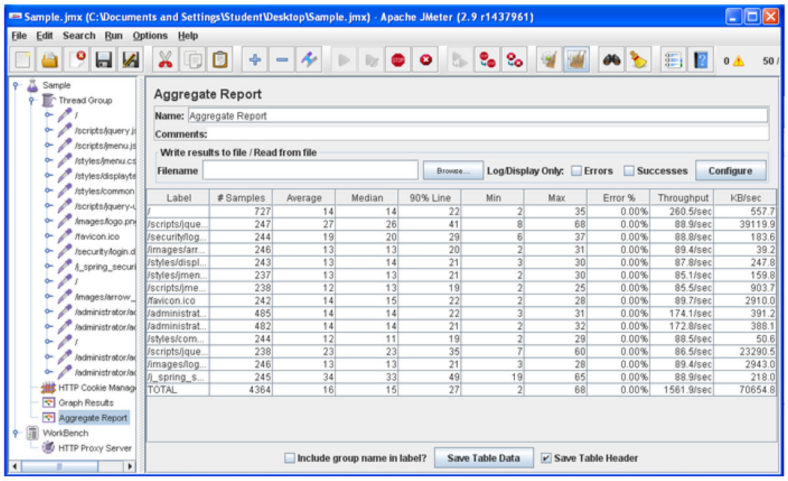
Let’s now report on aggregate reports.

Add an aggregate report



Graph reports are very good to have an overall picture of your throughput, but you also need an aggregate report so that you can inspect the details. To add such a report click on “Add > Listener” and select “Aggregate Report”.

A sample aggregate report



UNIVERSIDAD D SEVILLA 58

An aggregate report shows the following information per request (times are measured in milliseconds):

- Label: it refers to a URL.
- Samples: it's the number of times that the corresponding URL's been requested.
- Average: it's the average time taken to serve a request to the corresponding URL.
- Median: it's the median time taken to serve a request.
- 90% line: it's the 90th percentile of the time taken to serve a request.
- Min: it's the minimum time that a request took.
- Max: it's the maximum time that a request took.
- Error %: it's the percentage of HTTP errors caught while requesting a URL.
- Throughput: it's the average number of request to a URL that are served per second.
- KB/sec: it's the average number of KiB/second that are sent to the client when serving a request to a URL.

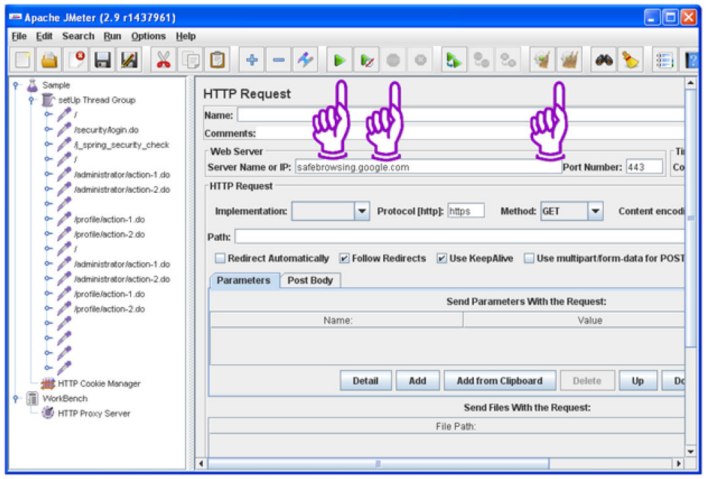
It's important that you understand these measures, since they can provide an accurate overall picture of how your system's performing while you run your tests.

Finally, it’s time to run your test cases.

Let’s try your test case

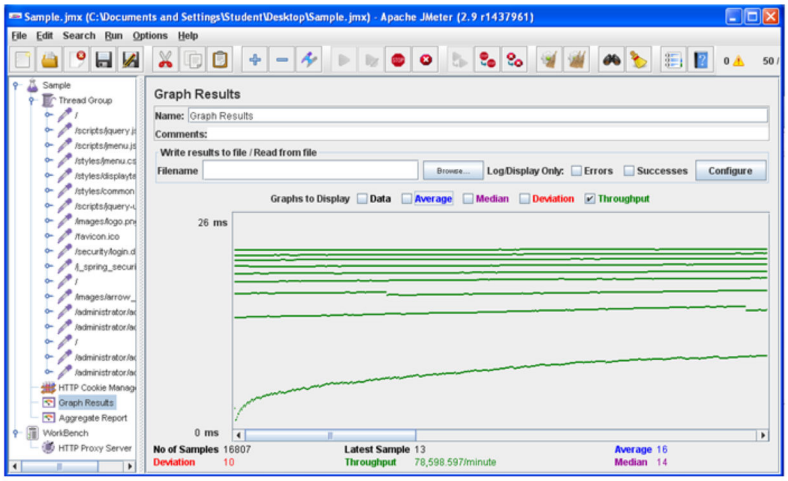
Cross your fingers! Very likely, this is the first time that your system’s serving several concurrent users!

Learn these buttons



jMeter provides the typical green triangle button to start running a benchmark, a red stop button to stop it, and a clear button to clear the reports. It's important that you clear your reports whenever you restart a test case since, otherwise, the new results will be added to the previous ones.

Check the graph report

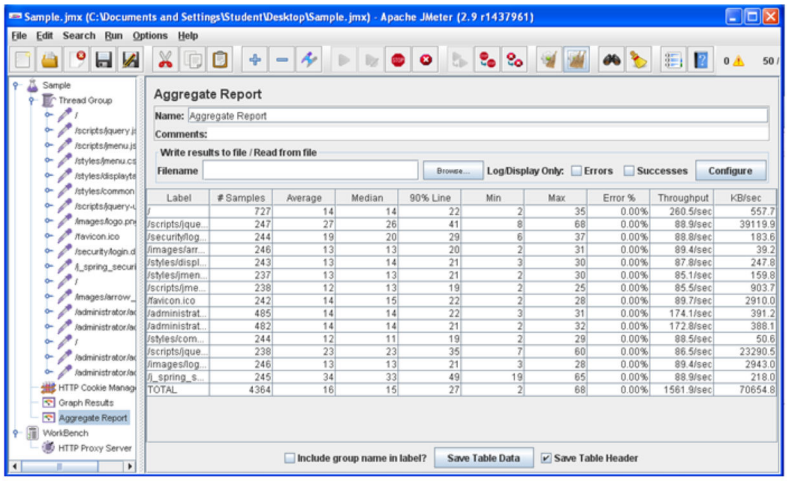


UNIVERSIDAD D SEVILLA

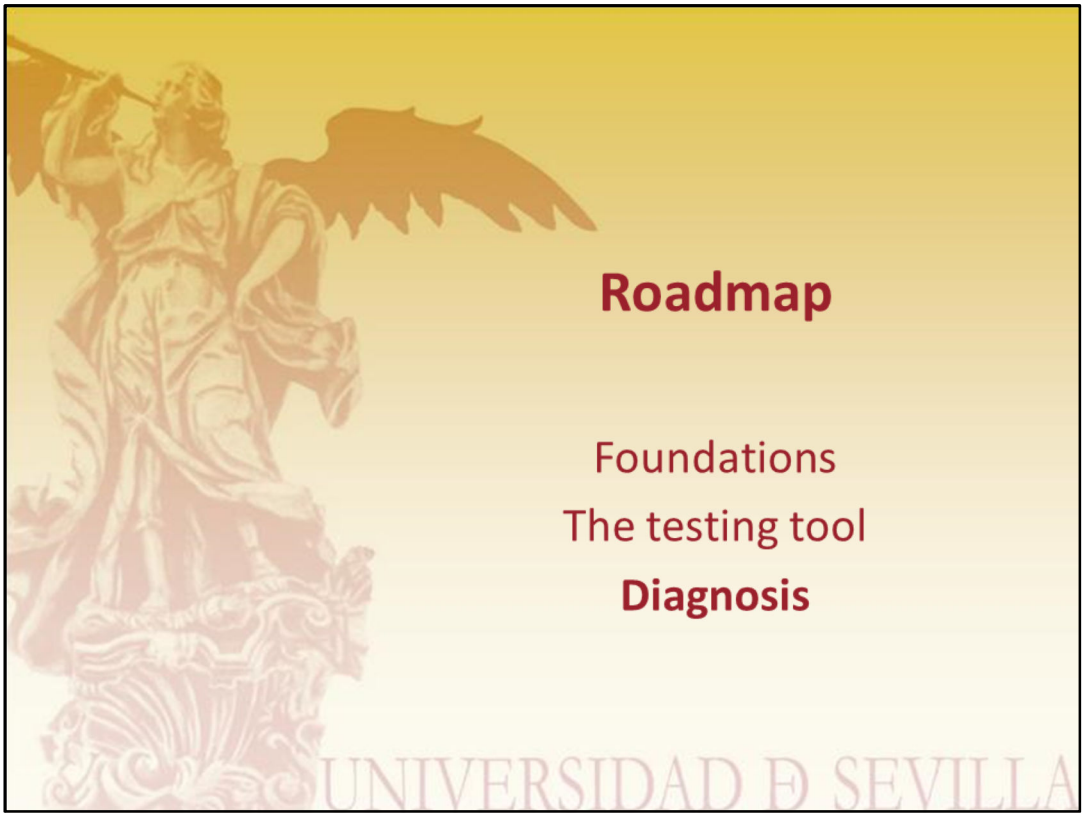
62

Click on the “Graph Results” report and you should see something very similar to this slide. The throughput is typically very small on the first few interactions and then it increases as the caches of your system get filled.

And also the aggregate report



Take also a look at your aggregate report and how the figures evolve as you test's executed.



Let’s finally report a little on diagnosis.

What's diagnosis about?

Diagnosis is about finding the reason why your system can't handle more workload

UNIVERSIDAD Ð SEVILLA

65

In the context of performance testing, diagnosis is about finding the reason why your system can't handle more workload.

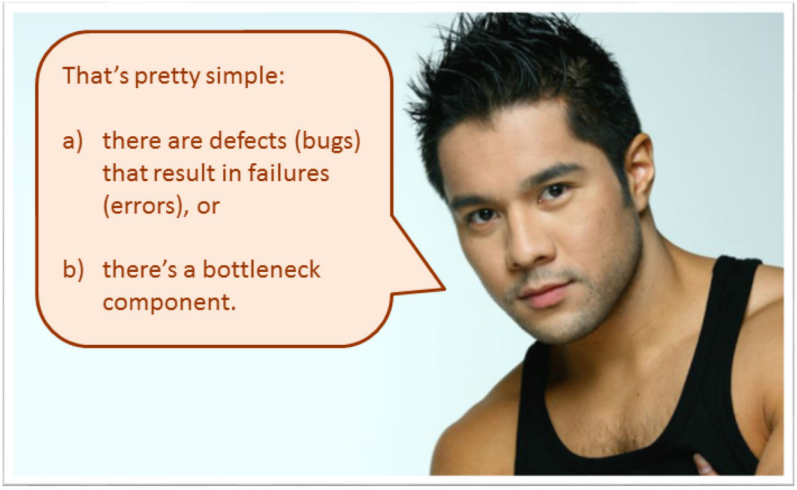
This is a good question!

UNIVERSIDAD Ð SEVILLA

66

Why would a computer not be able to handle enough workload is a good question… that deserves a good answer.

Basically and simply put



UNIVERSIDAD Ð SEVILLA

67

The answer's pretty simple: commonly a system cannot handle enough workload because there are defects (bugs) that result in failures (errors), or there's a bottleneck component.

Kinds of diagnosis

Failure diagnosis

Bottleneck diagnosis

UNIVERSIDAD Ð SEVILLA

68

So, it’s time to delve into the details regarding failure diagnosis and bottleneck diagnosis.

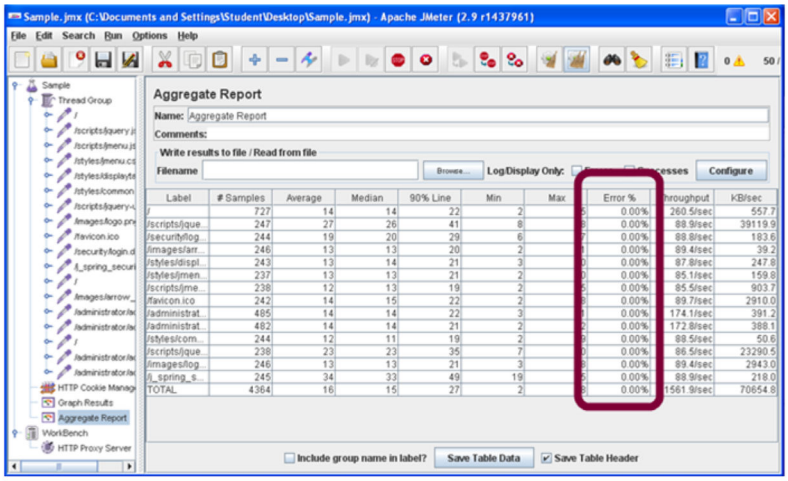
Kinds of diagnosis

Failure diagnosis

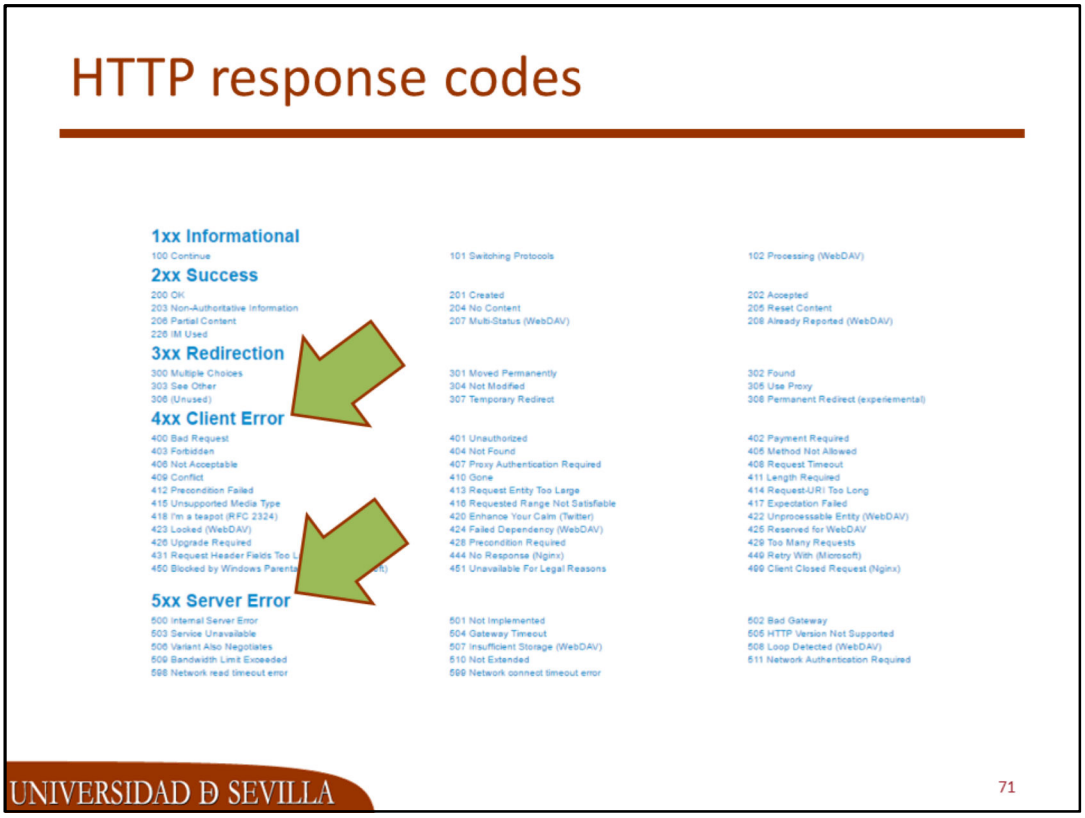
Bottleneck diagnosis

Let’s start with Failure diagnosis.

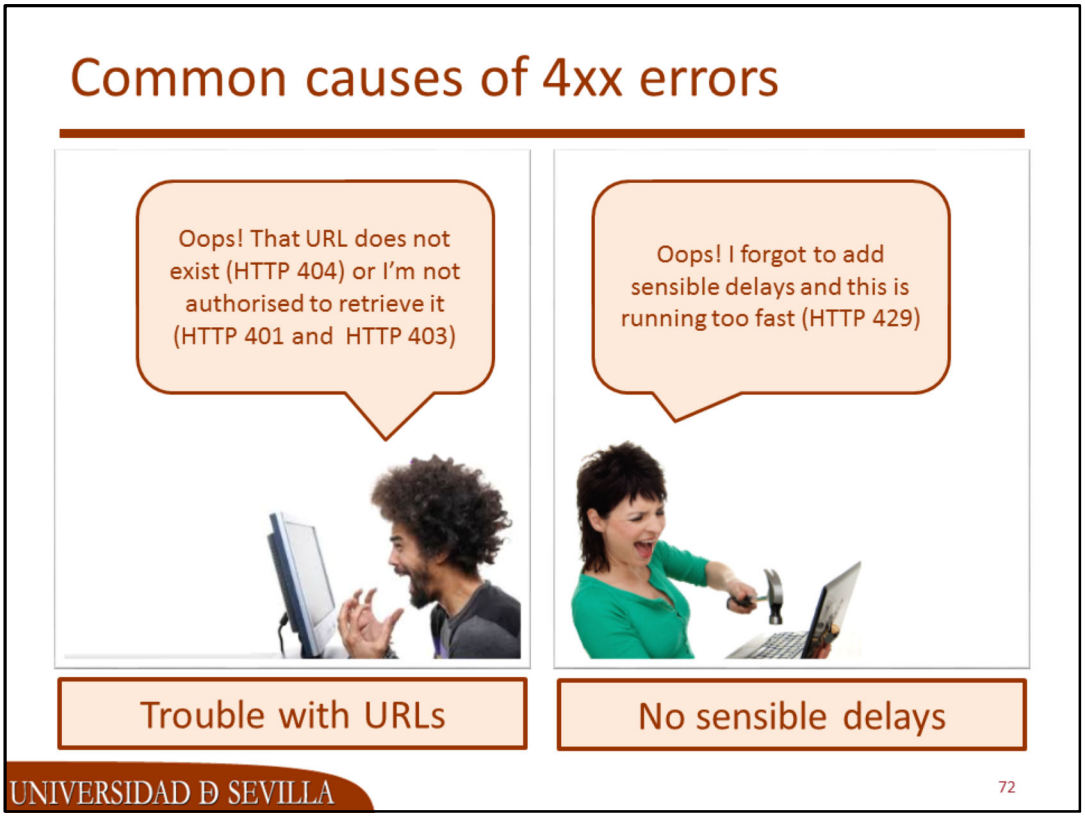
Check the HTTP errors



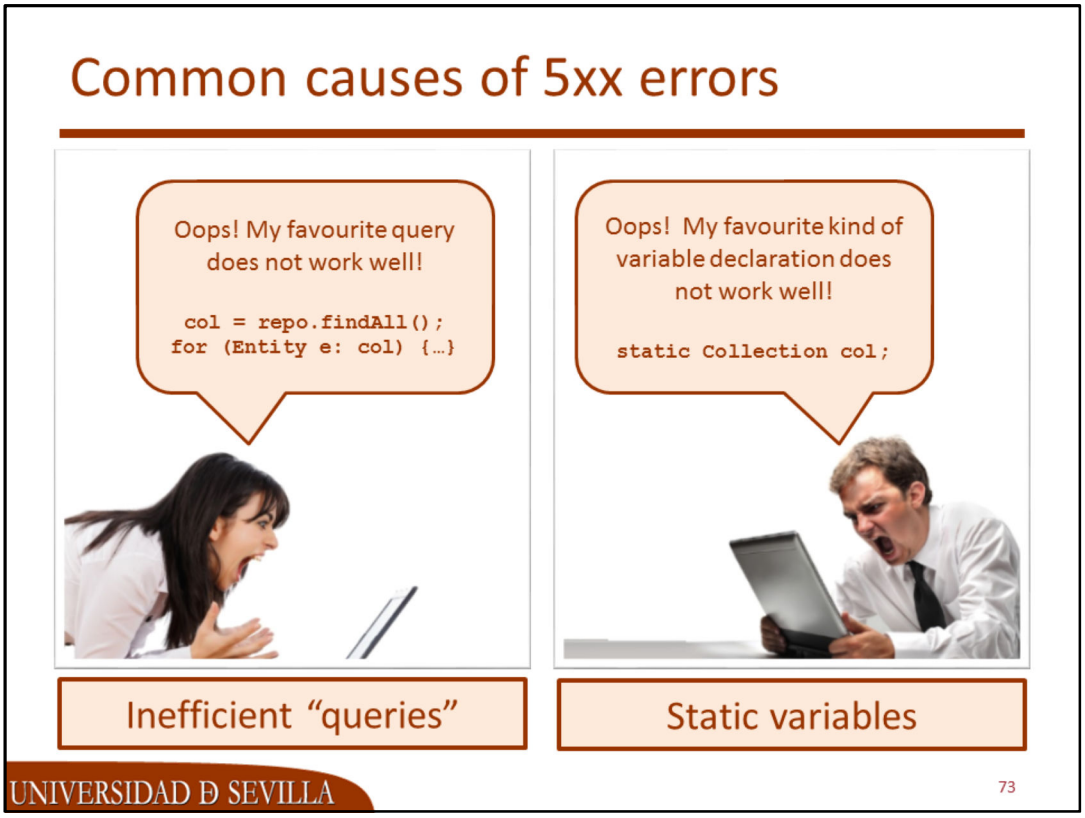
Recall that there's a column called "Error %" in aggregate reports. It measures the number of HTTP errors that are caught by jMeter.



Every request to a web server results in a response whose meaning is encoded in an HTTP response code. The codes are grouped into the 1xx series (informational), the 2xx series (success), the 3xx series (redirection), the 4xx series (errors due to the client), and the 5xx series (errors due to the server). jMeter counts an error when it gets an HTTP response code in the 4xx or the 5xx series.

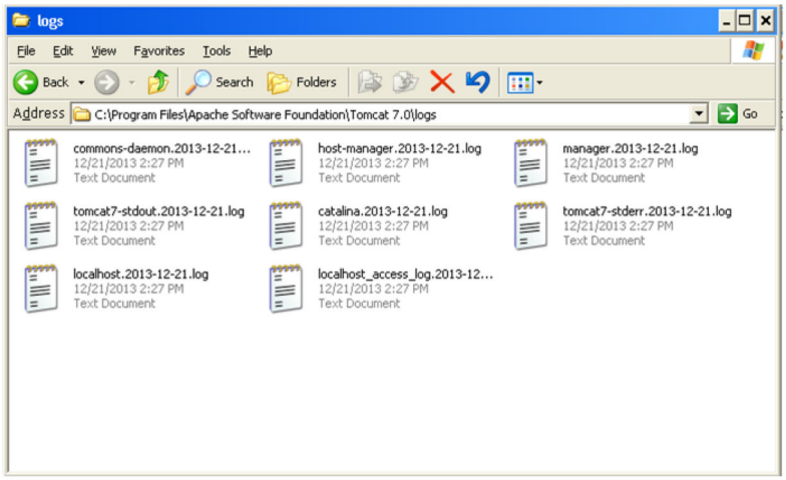


In this slide, we summarise the most common causes of 4xx HTTP errors. First off, you might have make a mistake regarding a URL, which might not exists in the server (HTTP error code 404) or the client might not be authorised to retrieve it (HTTP error codes 401 and 403). Second, if you forget to add sensible human delays to your scripts, then the server will likely reply with an HTTP error code 429 to let you know that your request rate is far too high.



In this slide, we summarise the most common causes of 5xx HTTP errors. First off, there are many students who do not make the most out of the JPQL language; they are not proficient enough with it and retrieve large collections of entities and process them in memory; that may work with a relatively small database as long as there are not many concurrent users; as the database grows in size and the users grow in number, this solution may very easily lead to not-enough-memory exceptions. Next, there are many students who like to use static variables to store data; such variables work well in a single-user setting, but they don't work well in a multi-user setting; as soon as the performance tests start simulating a few concurrent users, the race conditions that happen when accessing static variables typically result in a disaster.

Diagnosing HTTP errors



If you get an HTTP error, you must find the reason why… and there's only one way: you have to search for Tomcat's logs and go through them to find the problem. Recall that there's a folder called "logs" in Tomcat's installation directory. Tomcat's service logs information and error messages to the files in this folder. Please, search for the log files that correspond to the day on which you're working and analyse them.

Kinds of diagnosis

Failure diagnosis

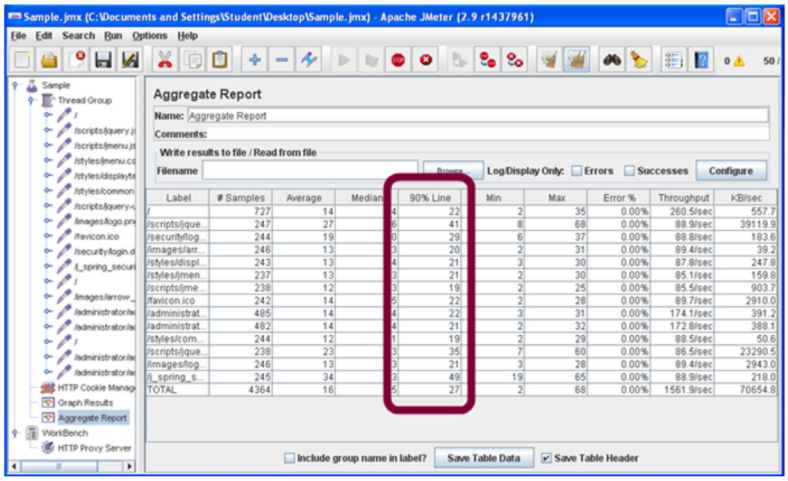
Bottleneck diagnosis

UNIVERSIDAD Ð SEVILLA

75

Let’s now report on how to perform diagnosis in order to find out what the bottleneck component of your system is.

Check the 90% line



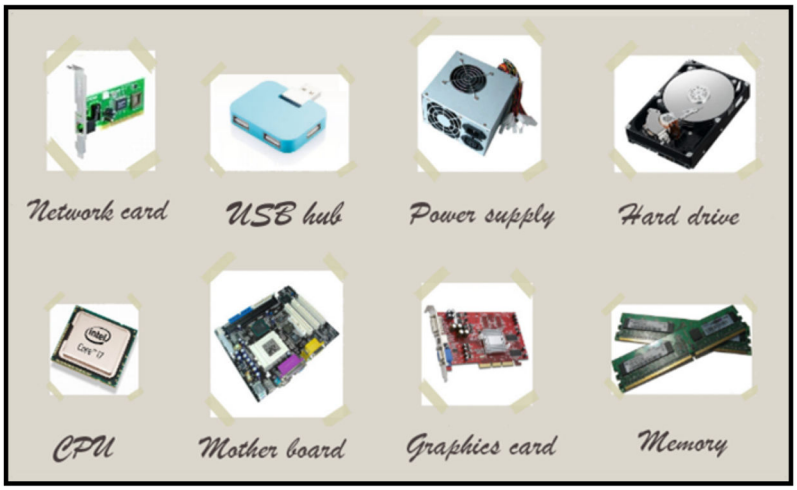
UNIVERSIDAD D SEVILLA

76

While you run your test cases, you must check that the average response time is not too high for the majority of users. Take a look at a measure called "90% Line" in your aggregate report; it measures the average time that 90% of the requests to a URL take to be served. For instance, the report in this slide shows that 90% of your requests have to wait for 22 ms to retrieve the index page of your web information system, 41 ms to retrieve a script, and so on. In total, the average time is 22 + 41 + 29 + … + 49 = 340 ms, that is, roughly 0.4 seconds. That's not a bad response time at all! But if the response time becomes too high, then you must diagnose your system to find its bottlenecks out. Keep reading, please.

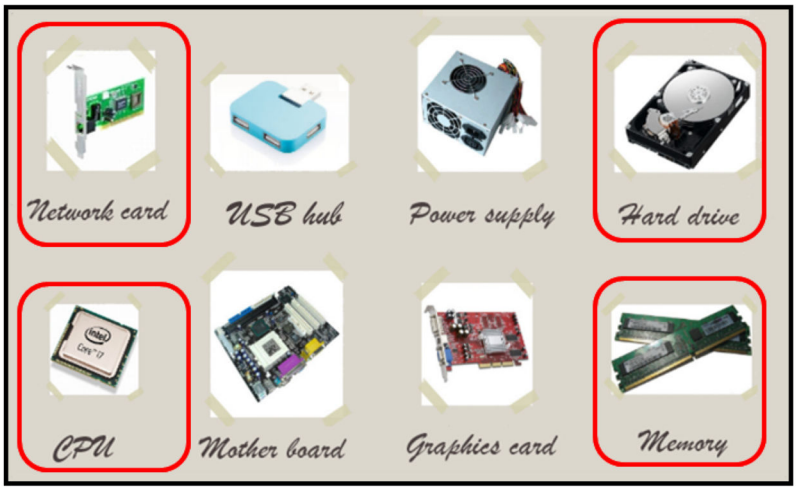
NOTE: note that the last line of the report seems to aggregate the sum of timings, but it is not computed correctly. Obviously, 22 + 41 + 29 + ... + 49 is not 27 ms. Sorry, you know that technology is far from perfect and jMeter's not an exception! It doesn't compute the totals well.

Components of a computer



This slide shows the typical components of which a typical computer is composed: a network card, a USB hub, a power supply, a hard drive, a CPU, a mother board, a graphics card, some memory chips… OK, the list isn't complete. It's just an summary of some key components.

Typical bottlenecks



UNIVERSIDAD Ð SEVILLA

78

When a computer runs a web information system, there are a number of components that become typical bottlenecks. A bottleneck's a component that doesn't work fast enough; that is, it typically collapses while the other components are not fully used or even idle most of the time.

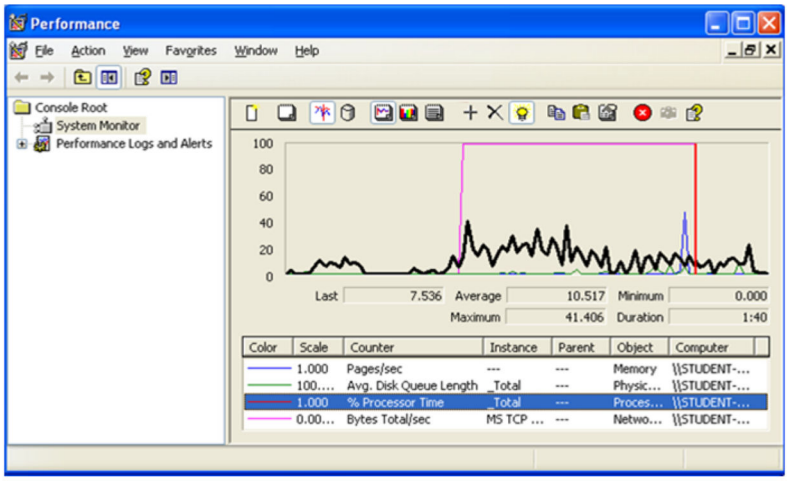
This is a good question

UNIVERSIDAD Ð SEVILLA

79

This is a good question: how can you discover which components of your system are bottlenecks.

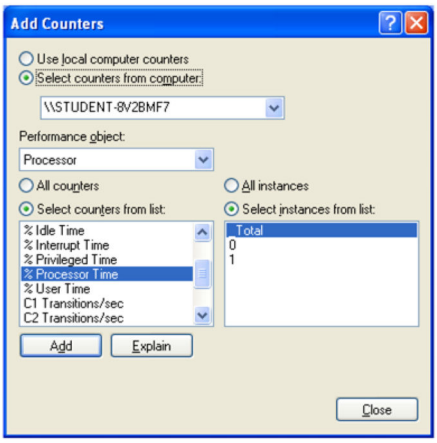
Use a performance meter



The answer's very simple: use a performance meter. In our pre-production configuration, the performance meter's an application that looks like in this slide. Open an administrator's shell and execute command "perfmon.exe". You need to add so-called performance counters to the performance meter so that you can monitor your system's activity while a performance test case is executed. In order to add performance counters, please, click on the "+" button in the button bar; unfortunately, there's no option in the main menu.

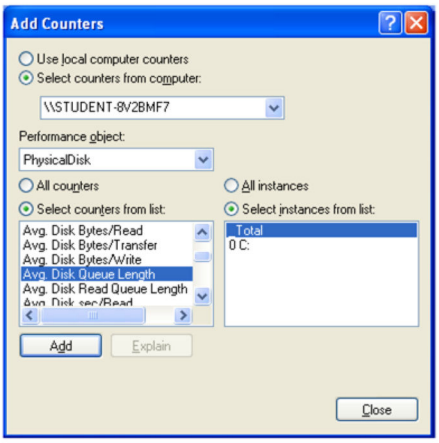
NOTE: there's not a universal performance meter. Please, find the most appropriate for your operating system.

Monitor your processor time



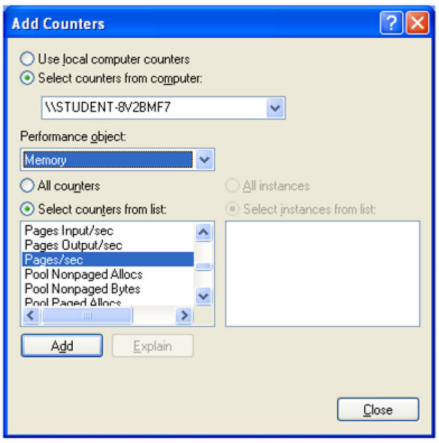
First, add a performance counter called “% Processor Time”. This counter measures the percentage of time that the processor spends to execute a user thread. It is the primary indicator of processor activity, and displays the average percentage of busy time observed during the sample interval.

Monitor your disk activity



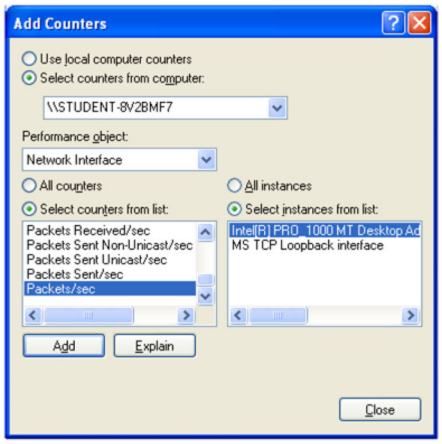
Now, add a performance counter called “Avg. Disk Queue Length”. It measures the average number of both read and write requests that were queued for the selected disk during the sample interval.

Monitor your memory faults



Next, add a performance counter called “Pages/sec”. This performance counter measures the rate at which pages are read from or written to disk to resolve hard page faults. This counter is a primary indicator of the kind of faults that cause system-wide delays. It includes pages retrieved to satisfy faults in the file system cache (usually requested by applications) and non-cached mapped memory files.

Monitor your network card



Finally, add a performance counter called “Packets/sec”. This counter provides an accurate estimate of the bandwidth that the network interface is delivering.

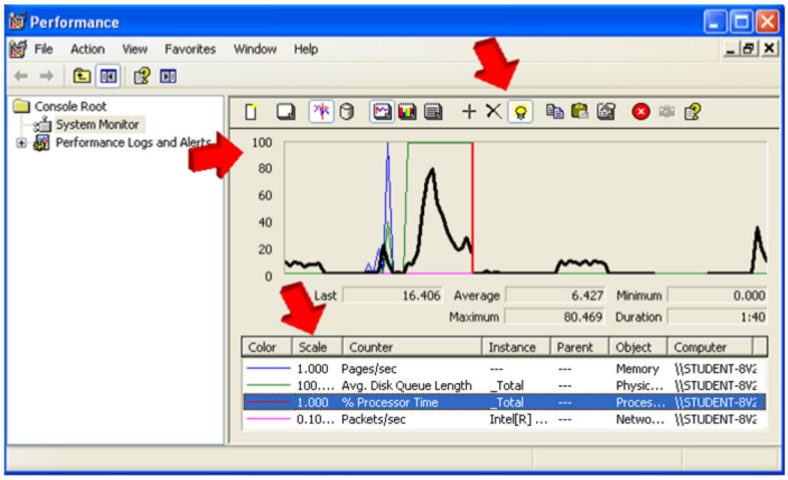
Nice to hear that!

UNIVERSIDAD Ð SEVILLA

85

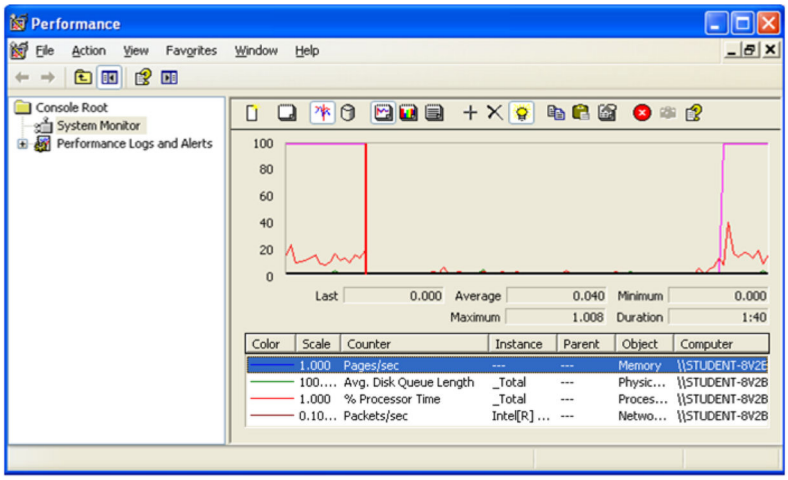
Typically, the previous performance counters are enough to diagnose a system and find its bottleneck. Let's now report on how to interpret the results that are provided by the performance meter.

The scale and the lamp



Realise that every performance counter has a scale, which is computed automatically; the scale is computed so that value 0 means that the corresponding component is idle and value 100 means that it's saturated. In order to identify the counters easily, there's a lamp button that we suggest that you should activate; when you activate this button, the performance counter that you select is represented as a bold, black line.

Are memory faults the problem?

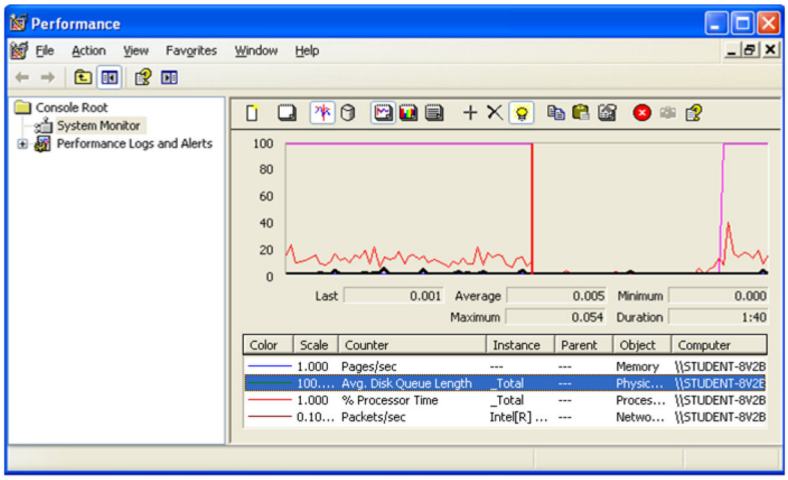


UNIVERSIDAD Ð SEVILLA

87

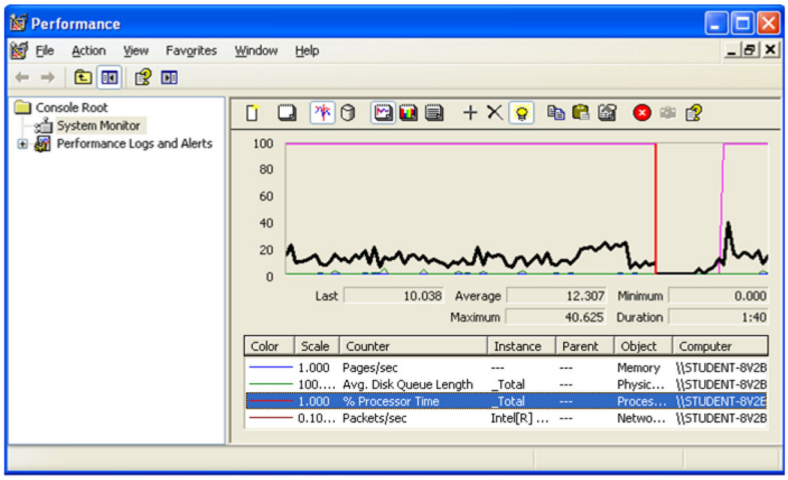
Assume that you're running your performance test cases and that you monitor your pre-production configuration. Is there a problem with your memory? Obviously not. Note that the "Pages/sec" counter is roughly zero all the time, which means that your memory is very, very far from becoming a bottleneck.

Is the disk the problem?



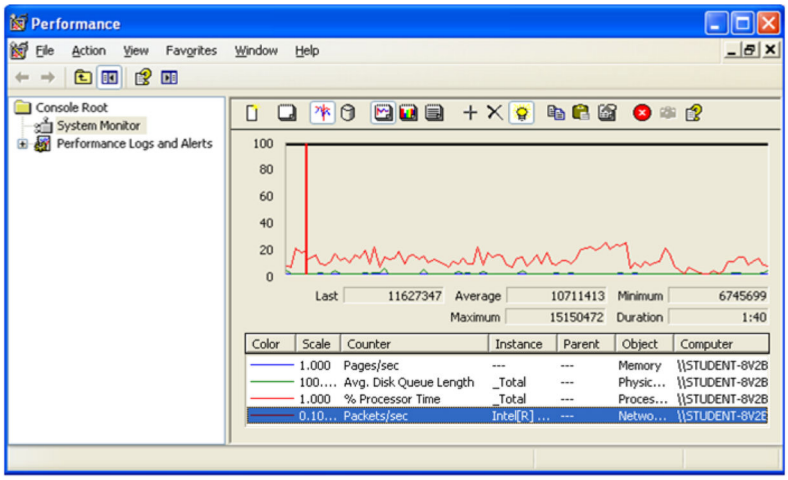
Is there a problem with your disk? No, not at all. Note that the "Avg. Disk Queue Length" performance counter is very close to zero all the time. That means that, other things equal, your disk might be able to serve a lot more requests.

Is the CPU the problem?



Neither seems it to be a problem with the CPU. Note that its workload's about 20%; in other words, it's very far from becoming a bottleneck.

Is the network card the problem?



UNIVERSIDAD D SEVILLA

90

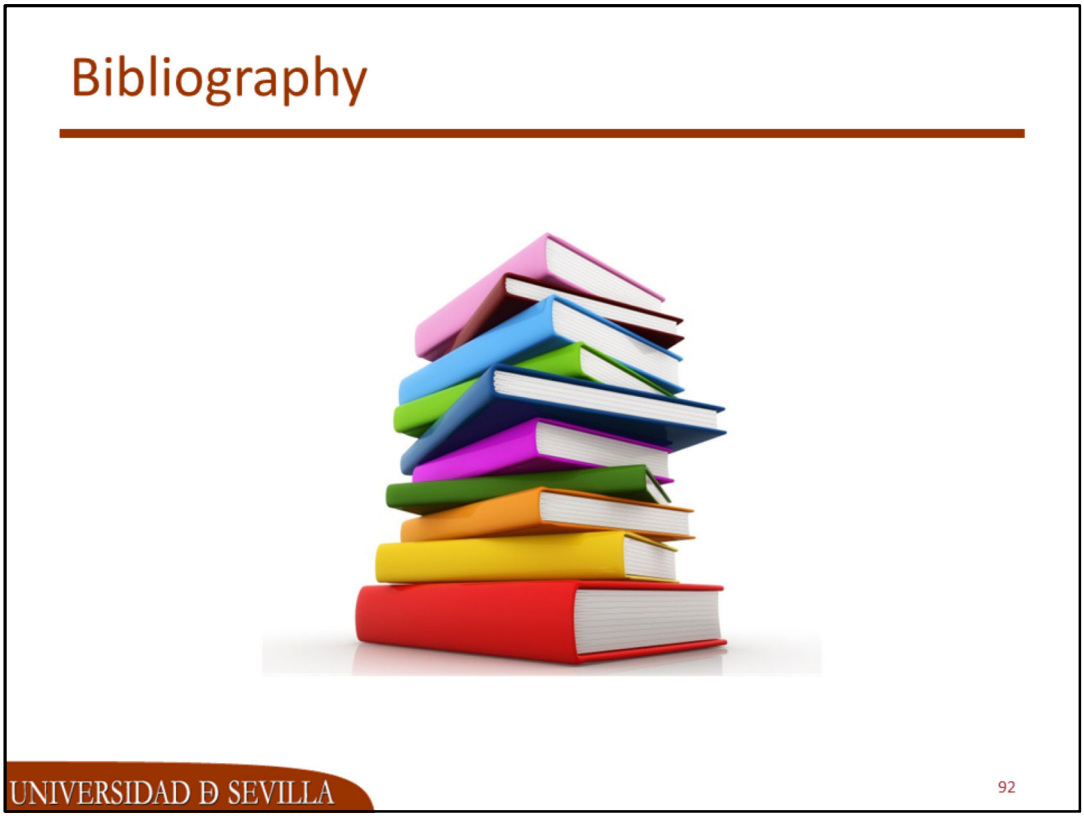
What about the network interface? Note that it's been running at 100% all the time; that means that this component can't handle the current workload because it's saturated and causes delays. The other components might deliver more throughput, but they can't because the network interface is not powerful enough to deliver data to them.

What's the conclusion?

UNIVERSIDAD Ð SEVILLA

91

Right, that's a conclusion: it seems that the bottleneck's your network card. In other words: replacing it with a more powerful one might boost your computer, which might help it serve a lot more requests per second. You don't have to update the CPU, the disks, or the memory, which are costly components, only your network card.



Unfortunately, there are not any electronic resources on performance testing available at the USE's library. For those of you who are seriously interested in this topic, we recommend that you should take a look at the following book:

Performance Testing With JMeter 2.9
Bayo Erinle
Packt Publishing, 2013

The official web site of jMeter is available at http://jmeter.apache.org; there you can find additional free learning resources, including a user manual and best practices. If you need some help with the HTTP codes, please, take a look at the list that is available at the official source https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html; if you don't think that source is easy to grasp, then take a look at http://www.restapitutorial.com/httpstatuscodes.html.

Thanks for attending this lecture!