Introduction to Performance Testing

Software Testing

Agenda

- What is performance testing?
- Attributes of Performance testing
- Types of Performance testing
- Goals of Performance testing
- Performance testing metrics
- Performance testing with Apache Jmeter

What is performance testing?

Performance testing is a non-functional testing performed to determine how a system performs in terms of responsiveness and stability under a particular workload.

Attributes of Performance testing

- **Speed** – how fast the system performs its intended functions
- **Scalability** - is the ability of a system, network, or process to handle a growing amount of work in a capable manner or its ability to be enlarged to accommodate that growth
- **Stability** – availability of the system without experiencing system failure / downtime

  Example of bad stability – some website crashes every 5 minutes
- **Reliability** - Ability of a computer program to perform its intended functions and operations in a system's environment, without experiencing failure (system crash)

  Example of bad reliability – when the system crashes it cause some data loss

Types of performance testing

Performance testing goals

- It can demonstrate that the system meets performance criteria.
- It can compare two systems to find which performs better.
- It can measure what parts of the system or workload causes the system to perform badly (“bottlenecks”)

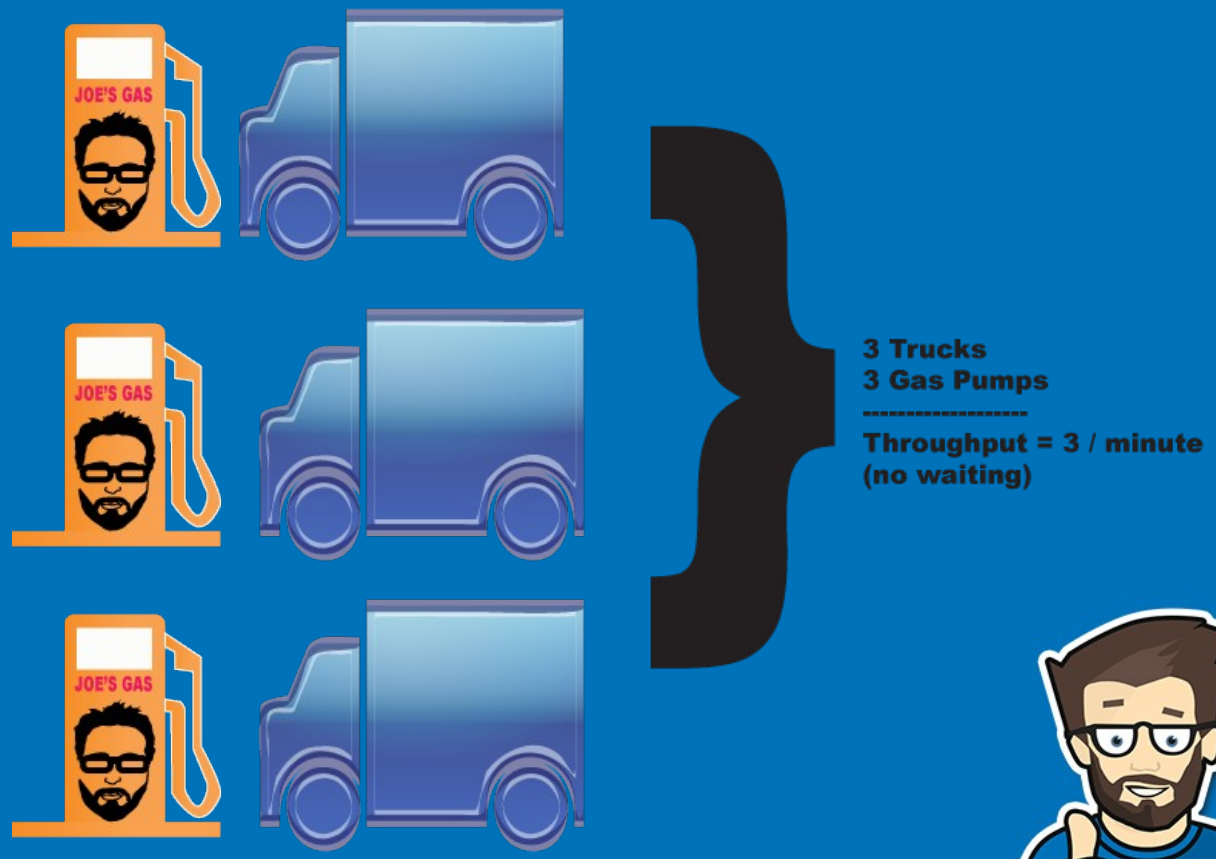
Performance testing metrics

- Concurrency/throughput (Transactions per seconds)
- Response time
- Error Rate
- Requests per Second
- CPU% usage
- Memory % usage

Concurrency/throughput

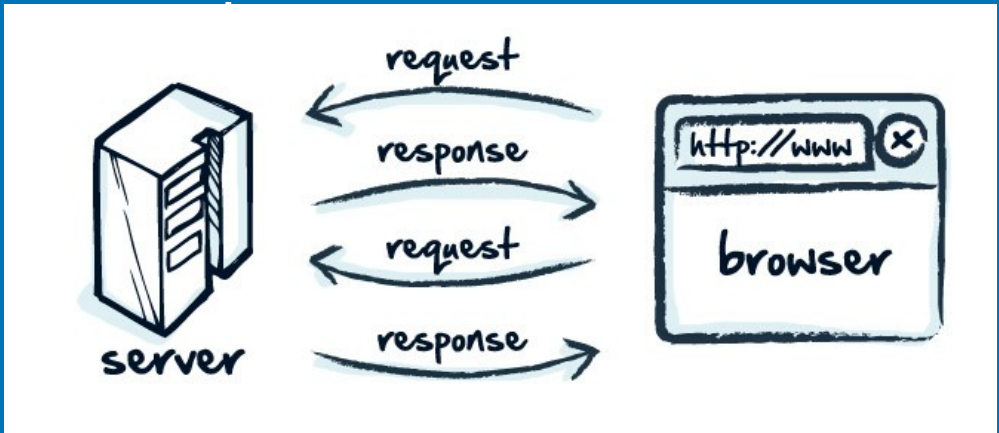
- **Concurrency** - largest number of concurrent system users that the system is expected to support at any given moment
- **Throughput** - the amount of transactions produced over time during a test

  Before starting a performance test it is common to have a throughput goal that the application needs to be able to handle a specific number of request per hr.



Response time

- Response time is the time spent by the server on responding to a



The response time increases proportionally to the user load.

Error rate

The Error Rate is the mathematical calculation that produces a percentage of problem requests to all requests. The percentage reflects how many responses are HTTP status codes indicating an error on the server, as well as any request that never gets a response.

Requests per seconds

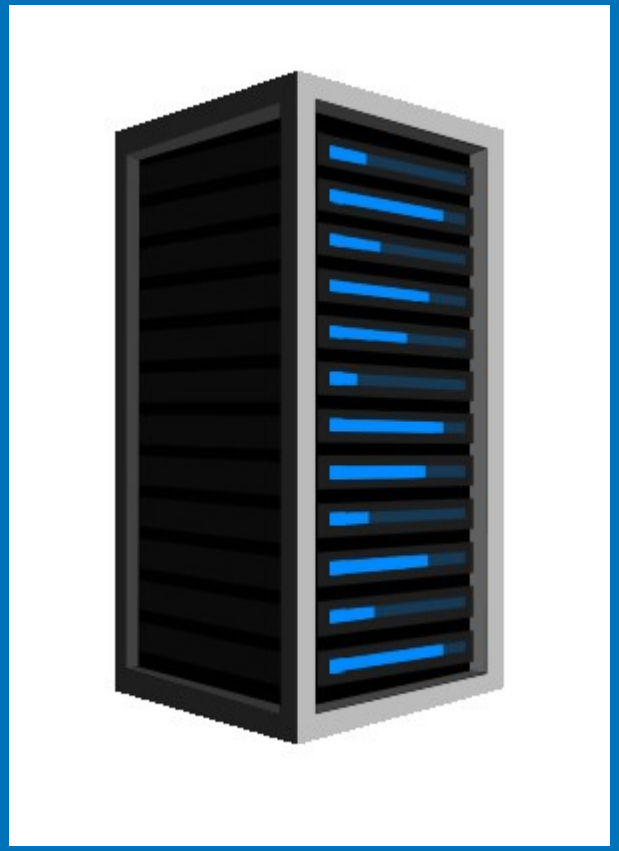
RPS is the measurement of how many requests are being sent to the target server.

RPS will be affected by how many resources are called from the site’s pages.

CPU usage and Memory usage

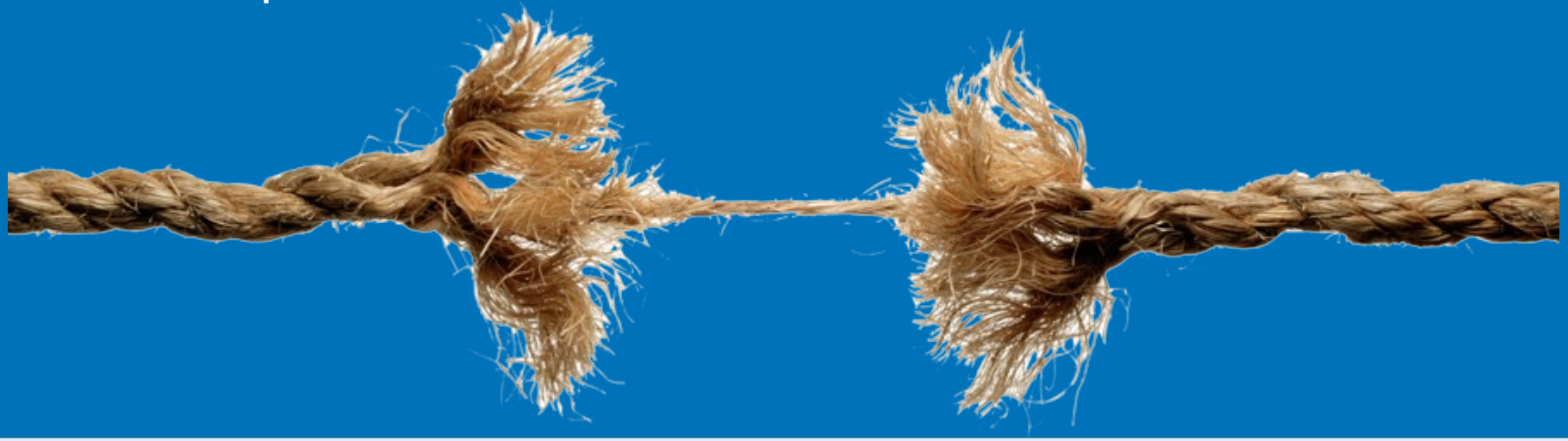
CPU % - The total CPU utilization as reported by the OS.

Memory % - The amount of VM memory that is available



Load testing

- Load testing is the simplest form of performance testing.
- Behaviour of the SUT under a specific expected load.
- Concurrent number of users on the app performing a specific number of transactions within the set duration
- The load testing shows how the system handles concurrent users effectively
- **Main metric:** Response time



Stress testing

- Stress testing is normally used to understand the upper limits of capacity within the system. This kind of test is done to determine the system's robustness in terms of extreme load and helps application administrators to determine if the system will perform sufficiently if the current load goes well above the expected maximum.
- **Main metric:** Response time and Throughput

Volume testing

- Volume testing is done against the efficiency of the application
- Huge amount of data is processed through the application
- Volume testing is done to analyze the system performance by increasing the volume of data in the database.
- **Main metric:** response time

Endurance/Soak testing

- Check that the system can withstand the load for a long or large number of transaction
- Performed with defined set of concurrent users for a prolonged period of time
- For example 8 to 10 hours or 2 to 3 days
- **Main metric:** memory usage

Spike testing

- Spike testing is done by suddenly increasing the number of or load generated by users - by a very large amount - and observing the behavior of the system. The goal is to determine whether performance will suffer, the system will fail, or it will be able to handle dramatic changes in load.
- **Main metric:** response time

Pre-requisites for performance testing

- **Non-functional requirements** should be available – will provide you with the expected Response time and required Concurrent user load, Scenarios to be tested and other performance attributes.
- **Dedicated Performance test environment** should be ready in place.
- Gather the environment details like App, Web and DB servers **configurations** and their capacity.

Performance testing tools

- Load Runner
- Apache JMeter
- The Grinder
- WebLoad
- NeoLoad
- LoadComplete
- Tsung
- BlazeMeter
- ..and many more

Performance testing with Apache JMeter

- Open source tool
- Supports:
  - HTTP/HTTPS
  - SOAP
  - REST
  - JDBC
  - LDAP
  - POP3(S), IMAP(S), SMPT
  - Etc.
- User friendly GUI compared to other tools
- Full multi-threading framework
- Test results can be captured in various format

Installing Apache JMeter

- System requirements:
  - JRE (Java Runtime Environment) installed
  - JAVA_HOME configured (check step-by-step guide)

- Download Apache Jmeter latest version (binary)
- Run Apache Jmeter from:
  <JMeter installation path>\bin\Jmeter.bat (for Windows OS)
  <JMeter installation path>/bin/Jmeter.sh (for Linux OS)

- **Example:**
  C:\Program Files\Jmeter\bin\jmeter.bat

Working with Apache Jmeter

Main elements of a Jmeter test plan:

- **Thread Group** – manages virtual users
- **Controllers** – manage requests to the server; two types – Samplers and Logical controllers
- **Listeners** - provide access to the information JMeter gathers about the test cases while JMeter runs
- **Timers** – manage different types of delays between requests, to better simulate real usage
- **Assertions** - Allow you to assert fact about responses received from server under test
- **Pre-Processors** – manage actions executed before the test
- **Post-Processors** – manage actions executed after the test

Full details available at Apache Jmeter manual

Running tests in Apache Jmeter

- Test can be run both from:
  - **Jmeter GUI (graphic user interface)** – best option when creating/editing the test plan and for debugging purposes
  - **CLI (command line runner)** – best option when running actual tests, especially with bigger loads (reduces resource requirements on the machine where the tests are executed)

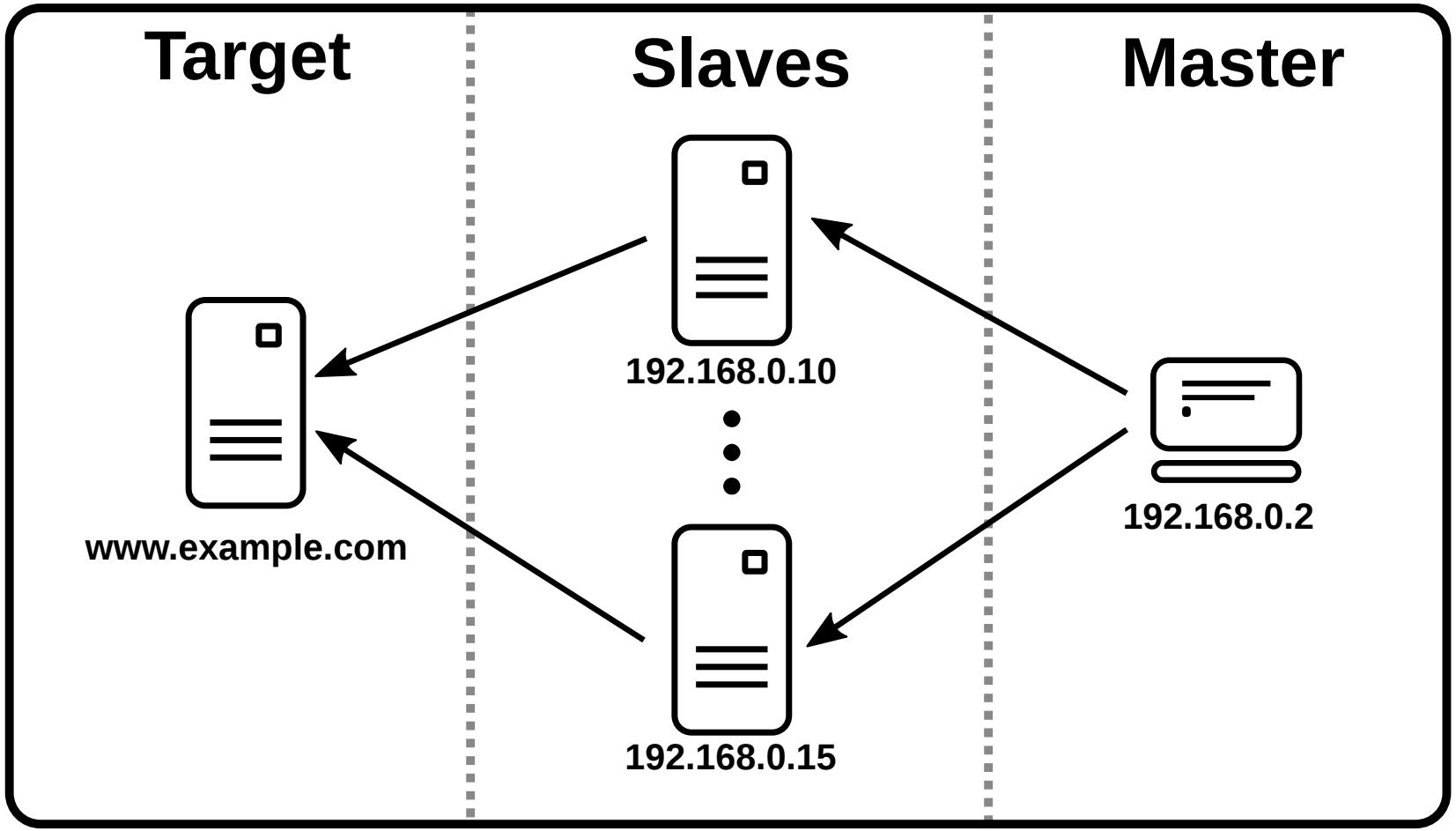
- To run tests from CLI:
  1) Open Windows / Linux terminal
  2) Navigate to Jmeter installation directory → bin folder *(where jmeter.bat or jmeter.sh is located)*
  3) Run following command (path to test.jmx will vary according to its location)
     jmeter -n -t <paht-to-test-plan>.jmx -l <path-for-log-file>.jtl

Running tests in distributed environment

- JMeter client machine may not able to simulate enough users to stress server.
- Control multiple machine to run JMeter without copying test samples to different machine.
- Configuration:
  - Copy same version of JMeter to different computer.
  - Add remote node IP in Jmeter properties file.
  - Run JMeter on remote machine using /JMETER_HOME/bin/jmeter-server (in command prompt)
  - Start JMeter GUI in host machine.
  - Select any test plan.
  - Go to Run >> Remote Start >> Remote IP Address.

SOFT ACADEMY

Distributed Test Environment



SOFT ACADEMY