Automation Testing

JMeter Distributed Testing

A Day in the Life of an Automation Test Engineer

Alex learned about the JMeter tool, which is used for a variety of non-functional testing.

Now, he is coming across certain situations where he noticed that his organization's e-commerce site faces a huge level of traffic most of the days. He knows that this issue can be resolved using distributed testing in JMeter.

In this lesson, he will learn about how to perform distributed testing in JMeter.

Learning Objectives

By the end of this lesson, you will be able to:

- Understand in which scenarios distributed testing in JMeter should be implemented
- Learn about the steps required to perform for working with distributed testing in JMeter
- Analyze the limitations that a user faces while doing distributed testing in JMeter

What Is Distributed Testing in JMeter?

Requirement of Distributed Testing in JMeter

If users run load tests locally on Apache JMeter, they can run only a limited number of users even if the computer has enough CPU and memory.

In such cases, the user needs to take the load tests to the next level and simulate larger numbers of concurrent users.

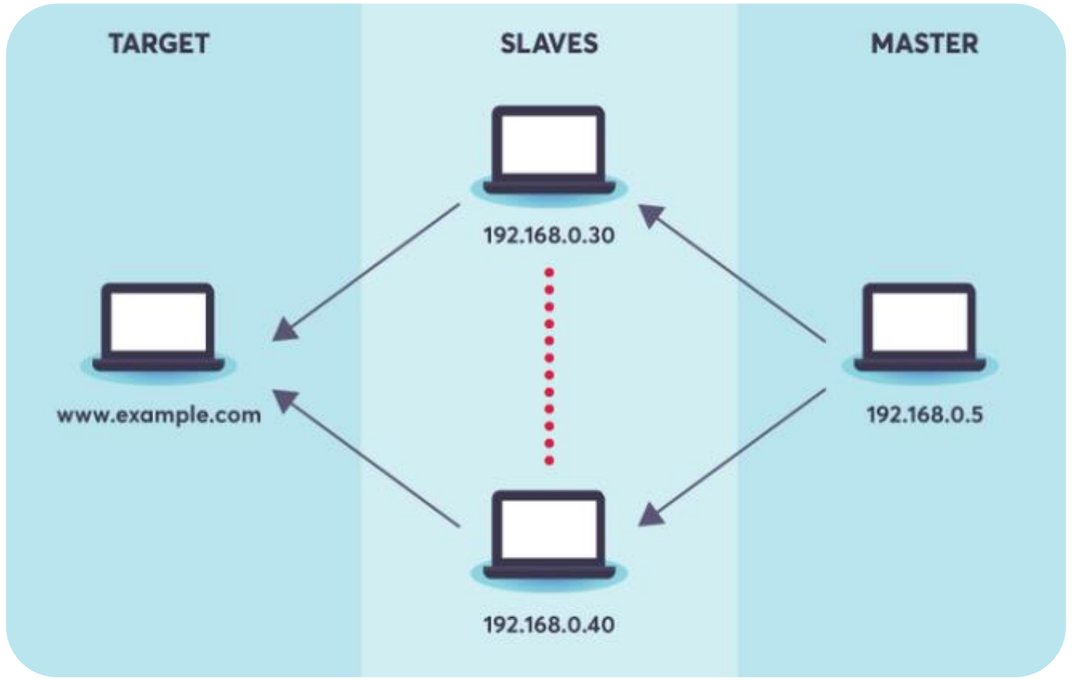
Requirement of Distributed Testing in JMeter

Relevant scenario: An organization is running an e-commerce site, it is completely normal that the company would expect a higher level of traffic on certain days, such as the Independence Day sale.

In such circumstances, distributed testing in JMeter is used.

Working with Distributed Testing in JMeter

Distributed testing in JMeter refers to a Master-Slave architecture where JMeter uses Java RMI (Remote Method Invocation) to interact with objects in a distributed network.



Working with Distributed Testing in JMeter

Distributed testing in JMeter enables having a local JMeter (master) that handles the test execution, together with multiple remote JMeter instances (slaves) that will send the request to the target server.

Before being able to run JMeter in a distributed way, the user needs to perform some useful steps.

Working with Distributed Testing in JMeter

Firstly, the user needs to possess multiple computers.

The user needs to get the JMeter Server running on each slave system.

Working with Distributed Testing in JMeter

To run the JMeter server on each slave system, the user must execute the **jmeter-server.bat** (**jmeter-server** for Unix users) that is located in the **jmeter/bin**.

After running, the result would be like this:

Working with Distributed Testing in JMeter

Using the same path, in the master system, the user should locate the **jmeter.properties** file.

The user would be editing this file and adding the IPs of all the slave systems that should be involved in the property ***remote_hosts***. The master and the slave systems should be located in the same subnet.

Working with Distributed Testing in JMeter

It is important for the user to remember that all the values must be separated by commas.

In the given example, there is only one slave system:

```
# Remote Hosts - comma delimited
remote_hosts=192.168.40.229
```

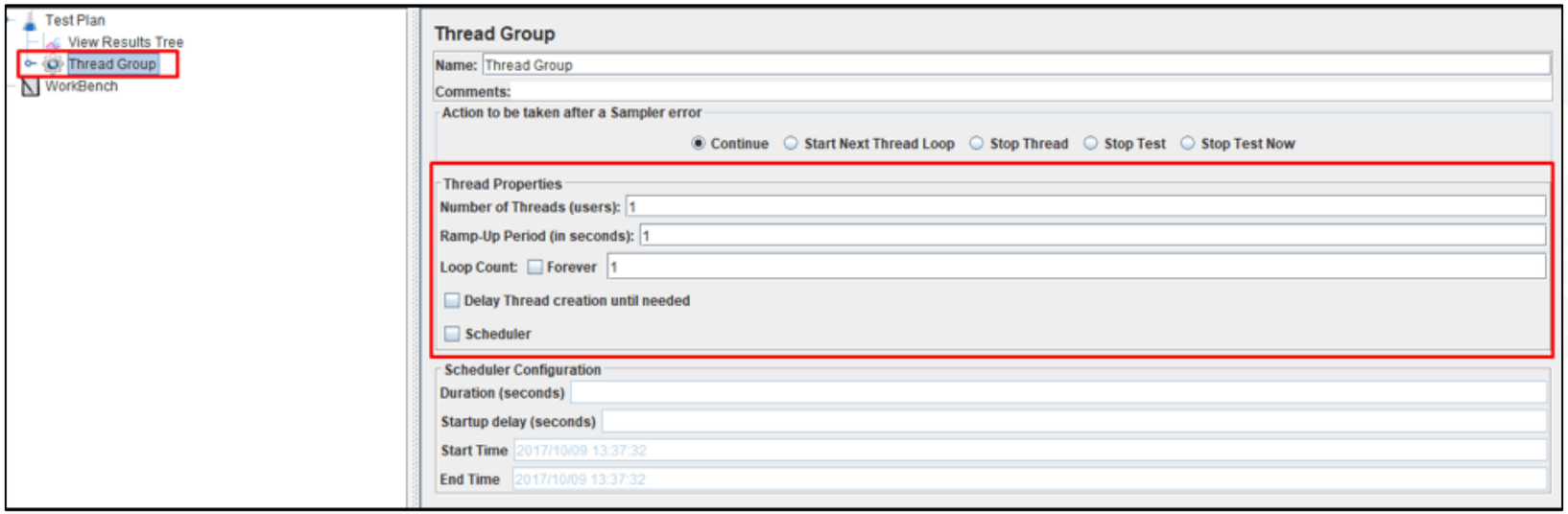
Working with Distributed Testing in JMeter

Once all these steps are done, the user can start JMeter and execute the required tests.

In this case, the execution in JMeter is done using GUI Mode.

Working with Distributed Testing in JMeter

In GUI mode, the number of users ramp up, and iterations should be configured in the master system's **Thread Group** as usual.



While running the test, these conditions should be replicated on each slave system.

Working with Distributed Testing in JMeter

After starting JMeter and loading the test, the user can go for two options:

- Configure through the master system by clicking on **Run-> Remote Start** and then select the slave system to be executed
- Configure through the master system by clicking on **Run-> Remote Start All** to start the test on all available slave systems

Working with Distributed Testing in JMeter

In this example, there is only one slave system and no difference between the two options.

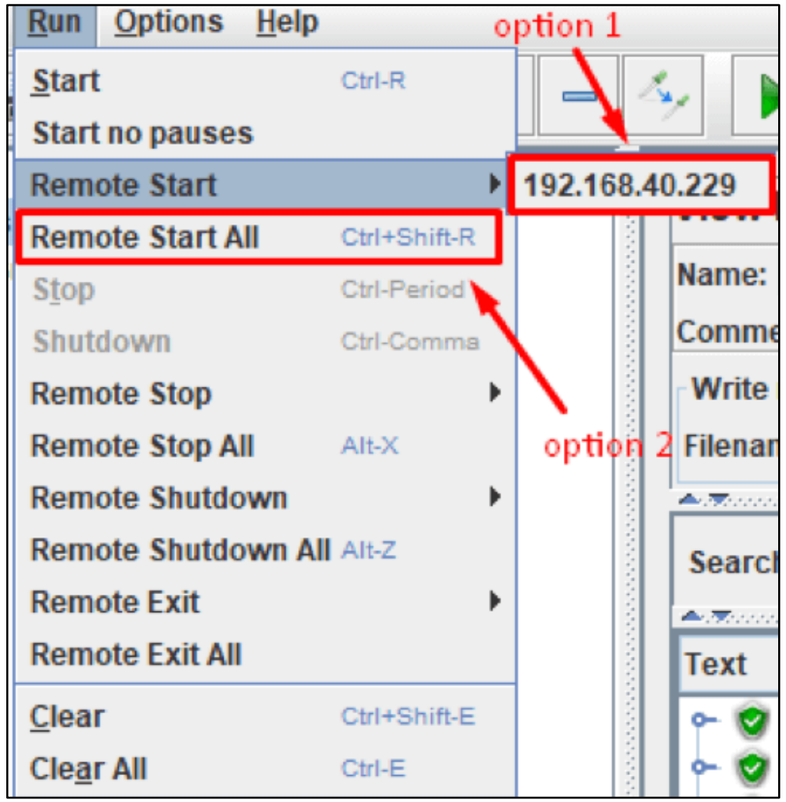


Image source: www.blazemeter.com

Working with Distributed Testing in JMeter

In this scenario, the user can search and view an article on an e-commerce site:

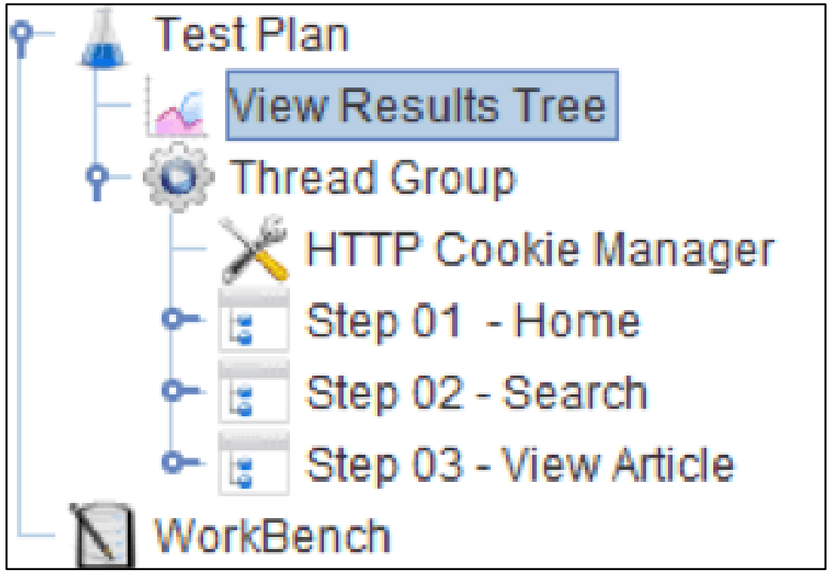


Image source: www.blazemeter.com

Working with Distributed Testing in JMeter

The **View Results Tree** listener is added so that the user can check whether the test was executed correctly or not.

When the user runs the test remotely, the user can locally see the test result on the **View Results Tree** listener. In the case of slave systems, the user can only observe the start and end time of the test.

Working with Distributed Testing in JMeter

When the user sees the test result locally, it looks like this:

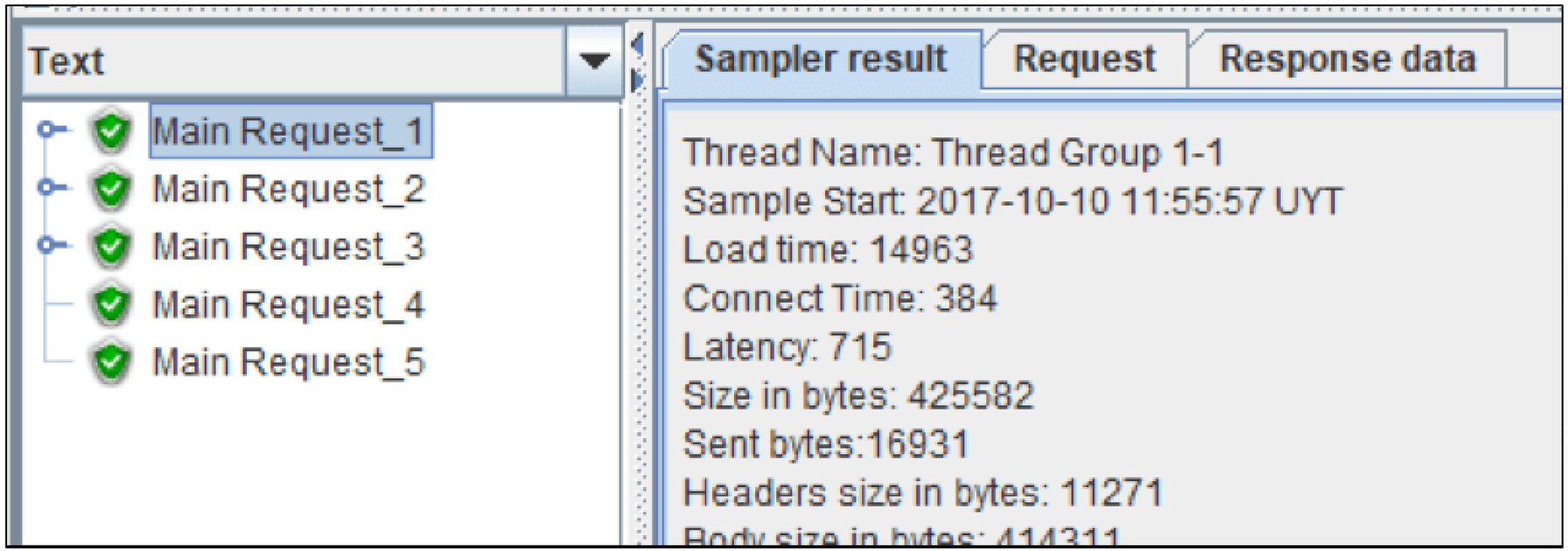


Image source: www.blazemeter.com

Working with Distributed Testing in JMeter

The test result remotely:

```
Starting the test on host 192.168.40.229
Finished the test on host 192.168.40.229
```

Image source: www.blazemeter.com

Working with Distributed Testing in JMeter

Each slave system executes the load tests with the conditions that the user sets in the master system.

Thus, users achieve a higher number of concurrent users and a higher load to the target server.

Working with Distributed Testing in JMeter

Therefore, if a user wants to distribute the load, the user must do it manually.

Example:
If anyone wants to reach 10,000 concurrent users and have 10 slave systems, then the test plan must have 1000 users, so that it ends up having 10,000 in total.

Limitations in Distributed Testing

If a user needs to simulate 25,000 concurrent users, the cost of maintaining all the systems is huge.

There may be a possible workaround for the maintenance issues to mount the architecture over Docker containers.

Limitations in Distributed Testing

The concept of distributing JMeter over Docker can be taken into consideration if there is a possibility to escalate the number of containers on demand.

Limitations in Distributed Testing

Secondly, while running JMeter in distributed mode, users face difficulties in handling .CSV files if there is parametrized data in their tests.

This problem occurs because users need to have separate files, and if they need to update them, then they must go to each slave system and make the modifications.

Key Takeaways

- Distributed testing in JMeter refers to a Master-Slave architecture where JMeter uses Java RMI to interact with objects in a distributed network.
- The View Results Tree listener should be added so that users can check whether the test was executed correctly or not.
- Each slave system executes the load tests with conditions that a user sets in the master system.
- Distributing JMeter over Docker can be taken into consideration if there is a possibility to escalate the number of containers on demand.