STANDARD OPERATING PROCEDURE

Setup Guide for JMeter Distributed Testing

1. **Purpose:** The purpose of JMeter Distributed Testing is to simulate a large number of concurrent users or requests to evaluate the performance, scalability, and reliability of a system under load. Distributed testing enables the use of multiple machines (known as "slave nodes") working together to generate a higher load than a single machine could handle, thereby providing realistic test scenarios for high-traffic applications.

2. **Scope:** This SOP outlines all essential steps, tools, and processes for setting up and conducting Distributed Testing, from installation and configuration to executing tests and analysing results for optimal system performance.

Document Details	
Reference Number	SOP-DT-001
Version	0.1
Effective Date	01-NOV-2024
Prepared By	Vinayak
Review By	Ankur Kumar
Approved By	Anupam Anand

Contents

Objective

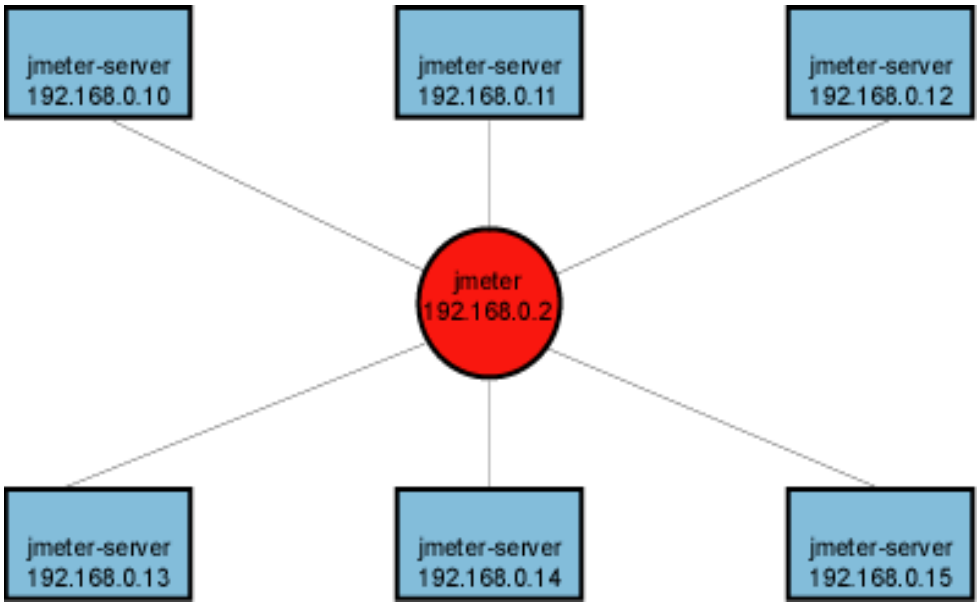
The objective of JMeter Distributed Testing is to simulate high user loads by distributing test execution across multiple machines. This approach helps evaluate system performance, scalability, and reliability under heavy load conditions while overcoming hardware limitations of a single machine.

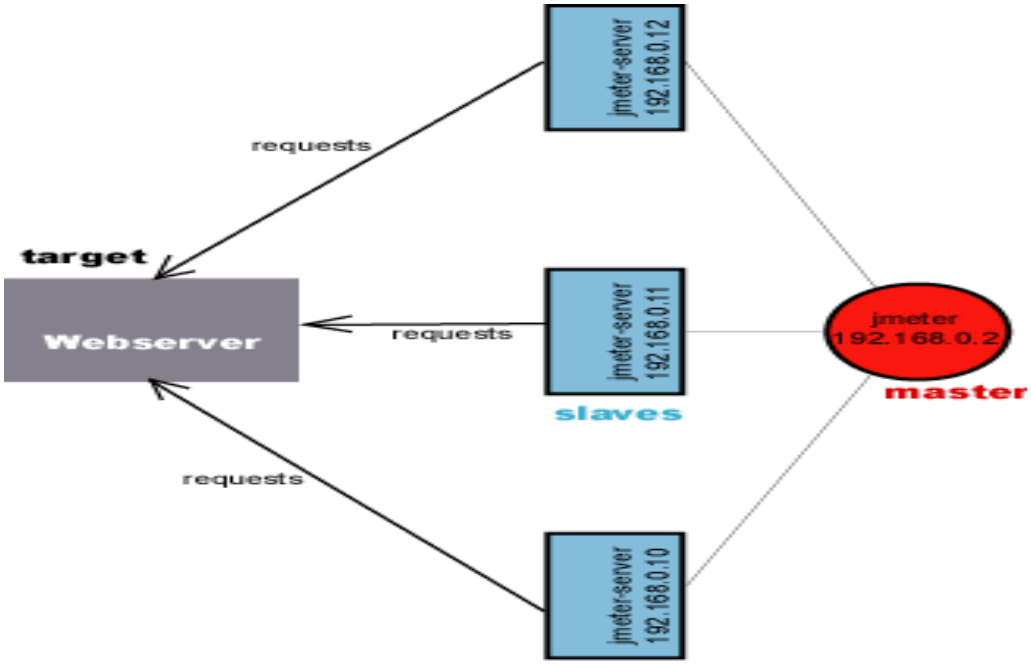
JMeter Distributed Testing

This Reports explains how to use multiple systems to perform stress testing. Before we start, there are a couple of things to check.

1. The firewalls on the systems are turned off.
2. all the clients are on the same subnet.
3. the server is in the same subnet, if 192.x.x.x or 10.x.x.x ip addresses are used. If the server doesn't use 192 or 10 ip address, there shouldn't be any problems.
4. Make sure JMeter can access the server.
5. Make sure you use the same version of JMeter on all the systems. Mixing versions may not work correctly.

Once you've made sure the systems are ready, it's time to setup remote testing. The way JMeter works is 1 master controller initiates the test on multiple slave systems.

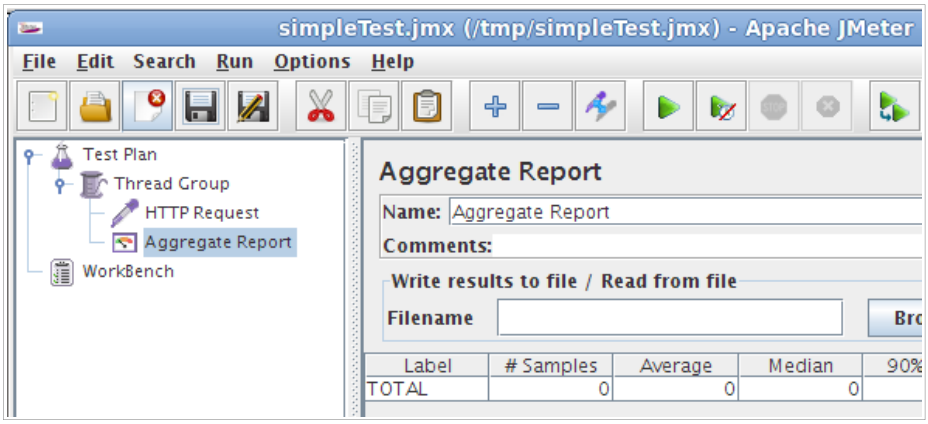




1. On master system acting as the console, open windows explorer and go to jmeter/bin directory
2. open jmeter.properties in a text editor
3. edit the line “Remote_hosts = Slave IP Address”
4. add the IP address. For example, if I have jmeter server running on 192.168.0.10, 11, 12, 13, and 14, the entry would like like this:

```
remote_hosts=192.168.0.10,192.168.0.11,192.168.0.12,192.168.0.13,192.168.0.14
```

5. Start j meter.
6. Open the test plan you want to use



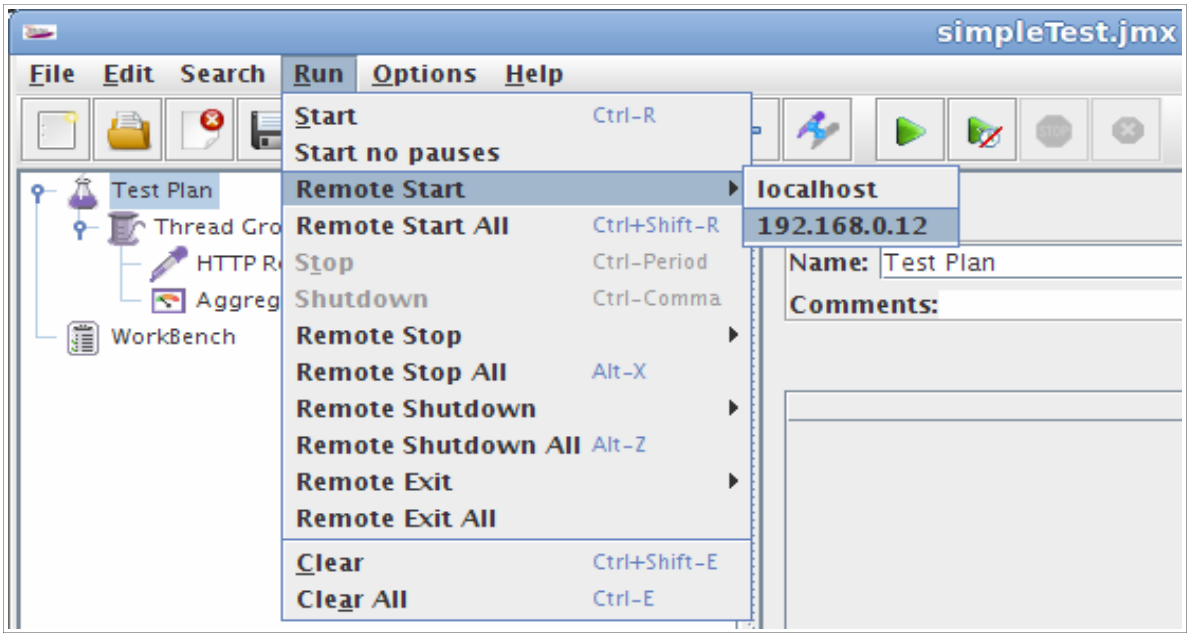
Starting the Test

At this point, you are ready to start load testing. If you want to double check the slave systems are working, open jmeter.log in notepad. You should see the following in the log.

Jmeter.engine.RemoteJMeterEngineImpl: Starting backing engine

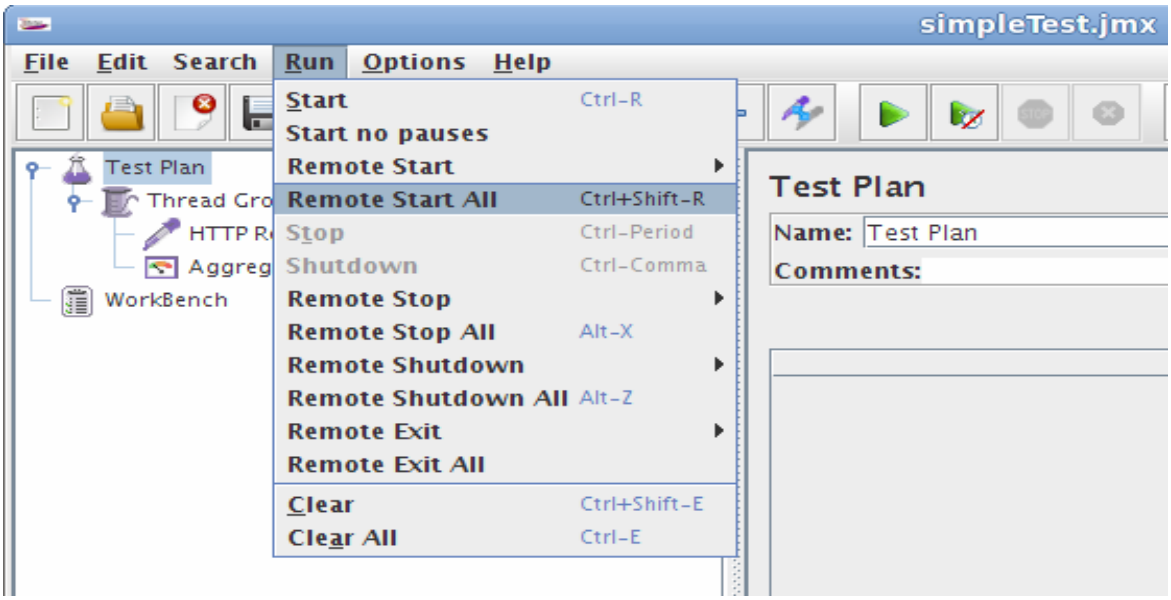
If you do not see this message, it means jmeter-server did not start correctly. For tips on debugging the issue, go to the tips section. There are two ways to initiate the test: a single system and all systems.

Start a single client



1. Click Run at the top
2. Select Remote start
3. Select the IP address

Start all clients



1. Click Run at the top
2. Select Remote start all

Slave Setup

1. Verify Network Connectivity

- Ensure both laptops are on the same network.
- Test connectivity between the master and slave:
    - From the master laptop, open the Command Prompt and type:
    - If the ping fails, check the network settings of both laptops (firewall, Wi-Fi, LAN, etc.).

2. Check JMeter Server on the Slave

- Ensure the JMeter server is running on the slave machine:
    - Open the terminal/command prompt on the slave machine.
    - Navigate to the JMeter bin folder.
    - Start the server:
    - Verify there are no errors while starting the JMeter server.

3. Firewall Settings

- The slave's firewall might be blocking the connection.
- On the slave laptop:
    - Open Windows Defender Firewall.
    - Allow JMeter's default port (1099) through the firewall:
        1. Go to Advanced Settings > Inbound Rules.
        2. Add a new rule to allow TCP connections on port 1099.
        3. Repeat for Outbound Rules.
    - Alternatively, disable the firewall temporarily for testing.

4. Validate jmeter.properties Configuration

On the Master:

- Open the jmeter.properties file in the master machine (located in the bin folder).
- Confirm that the remote_hosts property includes the slave's IP address:
- Save the file and restart the JMeter GUI or CLI.

On the Slave:

- Ensure the slave can bind to its IP address:
    - Add the following line to the slave's jmeter.properties file:
- Restart the JMeter server on the slave.

5. Validate RMI Configuration

- Both master and slave should use the same RMI settings.
- In the jmeter.properties file, ensure the following settings are consistent:This disables SSL for RMI, which can sometimes cause connectivity issues.

6. Check Java Version Compatibility

- Ensure both the master and slave laptops have the same Java version installed.
- To check the Java version:
- If there's a mismatch, update Java on both machines to the same version.

7. Restart Master and Slave

- Restart the JMeter server on the slave:
- Start the test from the master using the Remote Start option:
    - Go to Run > Remote Start > All.
    - Or use the command line:

8. Monitor Logs

- Check the logs on both the master and slave machines:
    - Master: jmeter.log in the bin folder.
    - Slave: The command prompt or terminal running jmeter-server.bat

Terminology

Before we dive into the step-by-step instructions, it's a good idea to define the terms and make sure the definition is clear.

Master – the system running J meter GUI, which controls the test
Slave – the system running j meter-server, which takes commands from the GUI and send requests to the target system(s)
Target – the webserver we plan to stress test