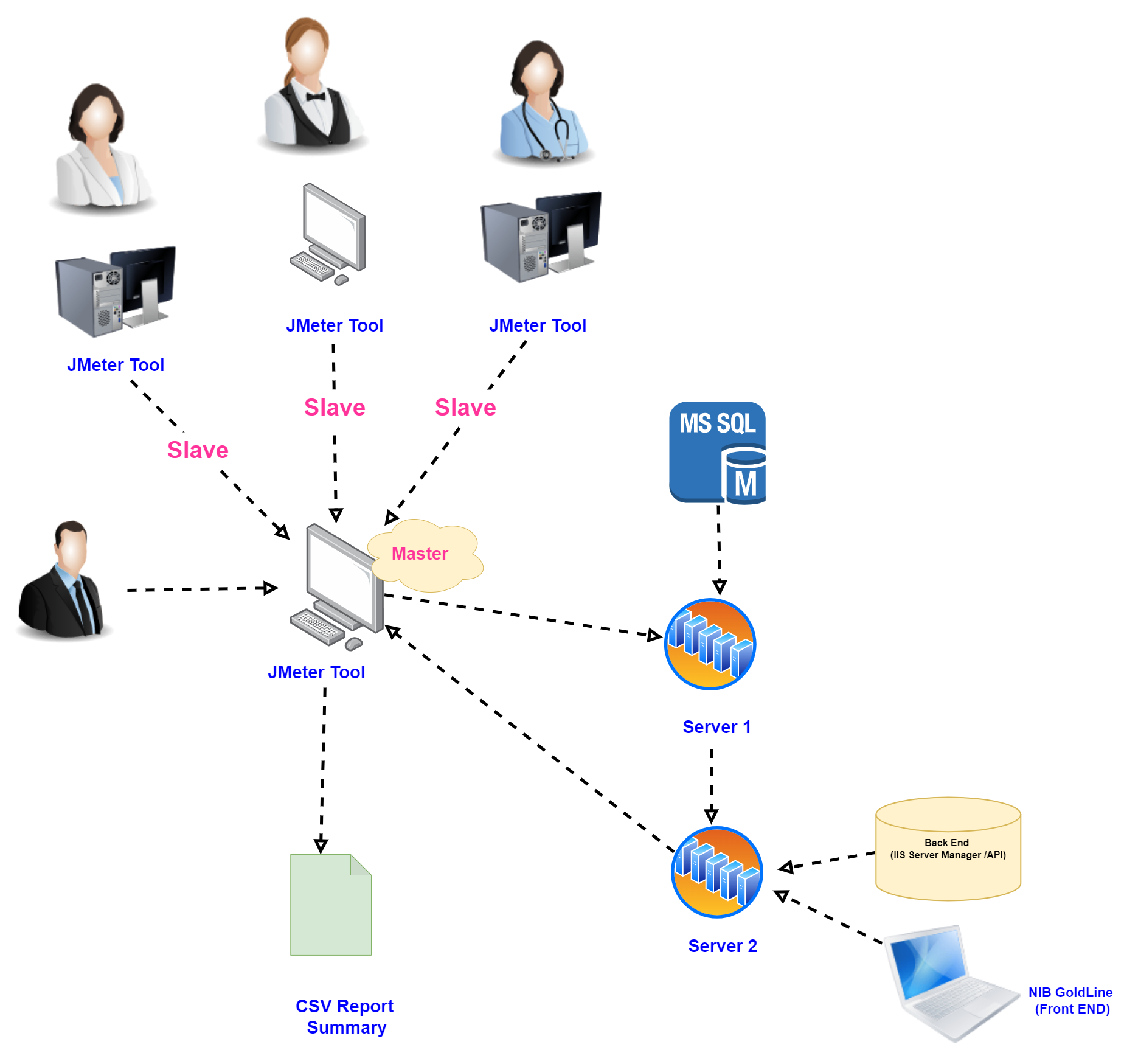
**Distributed Testing :-**

****

**Step 1:-**

Need to add IP Addresses in jmeter.properties of Master Machine

which located in C:\apache-jmeter-5.6.3\bin

remote\_hosts=192.168.1.74, 192.168.1.94

**Step 2 :-**

**Need to add 1099, 4445 and 60000 ports in a Inbound Rules**

**1. Verify Port 1099 is Listening on the Target Machine :-**

- **Check if the Port is Open:**

- On the target machine (`192.168.1.74`), open a Command Prompt and run:

```cmd

netstat -an | findstr :1099

- This will show if port `1099` is in a listening state. The output should look something like this:

TCP 0.0.0.0:1099 0.0.0.0:0 LISTENING

- If you don’t see this, it means nothing is listening on port `1099`,

and the issue might be with the JMeter configuration or the service that's supposed to be running on that port.

**2. Test Local Connection on the Server :-**

**- Test Telnet Locally:**

- On the server machine (`192.168.1.74`), try connecting to port `1099` locally to verify the service is accessible:

```cmd

telnet 127.0.0.1 1099

- If this connection fails, the problem is likely with the service configuration or it's not properly bound to port `1099`.

**3. Verify JMeter Configuration**

**-Check JMeter RMI Configuration:**

- Ensure that the `jmeter.properties` file on the slave machine (`192.168.1.74`) is correctly configured, particularly:

```properties

server.rmi.localport=1099

server.rmi.port=60000

server\_port=4445

- Also, ensure that `server.rmi.hostname` is set to the correct IP address (`192.168.1.74`).

**- Restart JMeter:**

- After making any changes, restart the JMeter server to apply the new configuration.

**4. \*\*Disable Firewall Temporarily\*\* :-**

**Test Without Firewall:**

- Temporarily disable the Windows Firewall on both the client and the server to determine if the firewall is indeed the issue:

1. Go to `Control Panel > System and Security > Windows Defender Firewall`.

2. Select `Turn Windows Defender Firewall on or off`.

3. Choose `Turn off Windows Defender Firewall` for both private and public networks.

- Try the Telnet command again:

```cmd

telnet 192.168.1.74 1099

- If it works with the firewall disabled, the issue is likely with the firewall configuration, and further adjustments may be needed.

**5. Check Network Connectivity:-**

**Ping the Server:**

- Verify basic network connectivity by pinging the server:

```cmd

ping 192.168.1.74

- If the ping fails, there might be a network issue preventing communication between the two machines.

**Check for Network Segmentation:**

- Ensure that there are no network segments, VLANs, or subnets that could be blocking traffic between the client and server.

**6. Check for Other Security Software :-**

**Third-Party Security Software:**

- If any third-party security software (like antivirus or network protection) is installed, it might be blocking the connection.

Temporarily disable it and test again.

**7. Test with a Different Port :-**

- **Test Another Open Port:**

- If you have another service running on a different port, try connecting to that port to see if the issue is specific to port `1099` or

if it's a broader problem.

**8. Check the Server's Binding IP :-**

**- Ensure the Service is Binding to the Correct IP:**

- Ensure that the JMeter server or the service listening on port `1099` is correctly bound to the IP address `192.168.1.74`.

It might be binding to `localhost` or another IP address instead.

**9. Network Security Policies :-**

**Check for Organization-wide Network Policies:**

- If you're in a corporate environment, there might be network policies or hardware firewalls preventing access to certain ports.

Check with your network administrator.

**Step 3 :-**

**Need to run this query on given path :-**

Follow this Path C:\apache-jmeter-5.6.3\bin and Run jmeter-server.bat

=>Need to this in all Machines like Master and Slave machines

Telnet is not directly accessible from the Control Panel in Windows. However, you can enable the \*\*Telnet Client\*\* feature through the Control Panel,

and then use it from the command line (Command Prompt).

**Steps to enable Telnet Client in Control Panel:**

**1. Open Control Panel:**

- Press `Win + R` to open the Run dialog.

- Type `control` and press \*\*Enter\*\* to open the Control Panel.

**2. Navigate to Programs:**

- In the Control Panel, click on \*\*"Programs"\*\* or \*\*"Programs and Features"\*\* (depending on the view).

- Then, click on \*\*"Turn Windows features on or off"\*\* (this option is located on the left panel).

**3. Enable Telnet Client:**

- In the list that appears, scroll down and find \*\*Telnet Client\*\*.

- Check the box next to \*\*Telnet Client\*\* and click \*\*OK\*\*.

- Windows will take a few moments to install and enable the feature.

**Using Telnet:**

Once Telnet Client is enabled, you can access it via the Command Prompt:

1. Open \*\*Command Prompt\*\* (`cmd`).

2. Type `telnet` followed by the server address or IP you want to connect to. Example:

telnet 192.168.1.1

Now Telnet is ready to use from the command line after being enabled from the Control Panel.

**Step 4 :-**

If your JMeter test takes \*\*15 minutes\*\* on the \*\*master machine alone\*\*, but when you add \*\*2 slave machines\*\*,

the total time becomes \*\*45 minutes\*\*, this suggests there is an underlying issue with the distributed setup. Normally,

adding slave machines should \*\*reduce the load on each machine\*\*, not increase the total runtime. Here are some potential reasons why this is happening:

**1.Master-Slave Communication Overhead**

**- Increased Coordination Time:**

The master machine coordinates with each slave during test execution.

If there is a significant delay in communication between the master and the slaves (due to network latency or synchronization delays), it can slow down the test as the master waits for results from the slaves before moving to the next step.

**- Synchronization Delays:**

If one slave is slower than the other, the test execution will be delayed because the master waits for all

slaves to finish their tasks before proceeding. This can increase the overall test duration.

**2. Network Latency or Bandwidth Issues**

**- Network Delays:**

If the slaves are located far from the master or there is poor network connectivity,

The delay caused by network latency can significantly increase the total execution time.

The master needs to send the test plan and receive the results over the network, and any delay in this process can impact the test.

**- Bandwidth Bottlenecks:**

If the amount of data being sent between the master and slaves is large (for example, detailed logs or large results),

network bandwidth can become a bottleneck, increasing the time taken to send and receive data.

**3. Unbalanced Load on Slave Machines**

**- Uneven Load Distribution:**

If the test load is not equally distributed across the slave machines,

one machine may be doing more work than the other, which can increase the total test execution time.

This can happen if one slave is handling more users or requests than the others.

**Different Hardware Configurations:**

If the two slave machines have different hardware specifications

(e.g., one is slower than the other), the slower machine could delay the entire test. The master waits for all slaves to finish before the test completes, so the slower machine becomes the bottleneck.

**4. Slave Machine Performance Issues**

**CPU or Memory Constraints:**

If either slave machine has insufficient CPU or memory resources to handle the load,

it may take longer to complete the test.

This could be due to resource constraints on the slave machines, causing them to process requests more slowly.

**- Disk I/O Bottlenecks:**

The slave machines may experience slow disk performance (e.g., if they're logging data or writing results to disk), which can slow down the test execution on those machines.

**5. Master Machine Processing Overhead**

**- Master Machine Overloaded:**

The master machine collects results from the slaves and processes them.

If the volume of results being returned is large, this could overload the master machine, slowing down the entire test.

The test may not finish until the master has processed all the results from both slaves.

**- Result Aggregation Delays:**

The master needs to aggregate results from the slaves in real time.

If this process is slow (for example, due to large result files or inefficient aggregation), it can delay the test completion.

**6. JMeter Thread Group Setup or Test Plan Design**

**- Sequential Execution in Test Plan:**

Check if there is any part of the test plan that is designed to run sequentially rather than in parallel across the slaves.

If the master or slaves are executing tests one after another rather than concurrently, this could increase the total time.

**- Incorrect Thread Group Configuration:**

Ensure that your thread groups are configured to properly distribute the load across the slave machines.

If the load distribution is imbalanced or not parallelized correctly, the test may take longer than expected.

**7. Garbage Collection and JVM Settings :-**

**- JVM Overhead on Slaves:-**

Each slave runs its own Java Virtual Machine (JVM). If the JVM on one or both slaves is not optimized (e.g., insufficient heap size),

garbage collection (GC) might be running more frequently, causing the slave to slow down. This can lead to delays in completing the test.

**- GC Delays:-**

If the slave machines are running into frequent GC pauses,

they may take longer to process requests and report results back to the master.

**Step 5 :-**

**How to Troubleshoot the Issue:-**

**1. Check Network Performance:-**

Ensure the master and slaves are on a reliable network. Test the latency and bandwidth between the master and the slave machines.

If there is high latency, consider moving the machines to the same network or closer to reduce delays.

**2. Monitor Resource Usage:-**

Use monitoring tools to track CPU, memory, and network usage on both the master and slave machines during the test.

Look for any bottlenecks or resource constraints that may be causing delays.

**3. Optimize JVM Settings:-**

Increase the heap size allocated to the JVM on both the master and slave machines.

This can help reduce garbage collection pauses and improve performance.

**4. Check Test Plan Configuration:-**

Review the test plan to ensure that all elements are running in parallel and not in a sequential manner.

Also, verify that the load is distributed equally across all slaves.

**5. Analyze Logs:-**

Check the JMeter logs on the master and slave machines for any errors or warnings that could indicate performance issues or misconfigurations.

**6. Simplify Test Data:**

If the master is collecting a lot of detailed results from the slaves, try reducing the level of logging or the amount of data being collected to see if this improves performance.

By addressing these potential causes, you can reduce the test execution time and ensure that distributed testing works efficiently.

**Step 6:- In ExcelSheet Report**

**To convert \*\*291,838 milliseconds\*\* into minutes, I followed these steps:**

**1. Convert milliseconds to seconds:**

**- Since 1 second = 1000 milliseconds, divide the milliseconds by 1000:**

291838 \, \text{milliseconds} \div 1000 = 291.838 \, \text{seconds}

**2. Convert seconds to minutes:**

**- Since 1 minute = 60 seconds, divide the seconds by 60:**

291.838 \, \text{seconds} \div 60 = 4.864 \, \text{minutes}

So, \*\*291,838 milliseconds\*\* equals approximately \*\*4.86 minutes\*\*.

**Step 7:- In ExcelSheet Report**

1. **In Timestamp Need to change this Format Cell in Custom Number like Below :-**

-> dd-mm-yyyy hh:mm:ss

1. **After that need to change the elapsed time milliSeconds to Seconds :-**

-> Select the Elapsed time Seconds the Select Column and Value Like R2/1000

**Step 8 :-**

To upload these Documents in a GIT

<http://192.168.1.229/HarshilModi/performance-testing> :-

**\*\*Need to run this Queries :-\*\***

1. git add.
2. git commit -m “comment”
3. git push origin develop

**Example :-**

The command you provided uses **Apache JMeter's non-GUI mode** for distributed testing. Let me explain the components of your command and how they work, along with documentation to match your scenario:

jmeter -n -t D:\Jmxfiles\All\All.jmx -l C:\apache-jmeter-5.6.3\bin\1160APIs1MinRamp5Distribute.csv -R 192.168.1.74,192.168.1.75

#### **Command Options Explained**

1. **-n**: Runs JMeter in **non-GUI mode**.
2. **-t**: Specifies the path to the **test plan file** (.jmx file) to execute.  
    In this case:  
    D:\Jmxfiles\03\_10\_2024\_Update\All\All.jmx
3. **-l**: Specifies the path to the **results file** where the test output will be saved.  
    In this case:  
    C:\apache-jmeter-5.6.3\bin\1160APIs1MinRamp5Distribute.csv
4. **-R**: Specifies the **remote servers** (comma-separated IP addresses) to run the distributed test on.  
    In this case:  
    192.168.1.74

### **How Distributed Testing Works :-**

#### **Setup for Master-Slave Configuration :-**

1. **JMeter Master Machine (Your Local Machine) :-** 
   * This is where you run the command. It coordinates the test execution across the remote slave servers.
2. **JMeter Slave Machine (Remote Server at 192.168.1.74):-**
   * This machine(s) executes the test plan using the configurations from the master.

#### **Steps to Configure Distributed Testing :-**

1. **Start Slave Servers on Remote Machines**

* On each slave machine, navigate to the JMeter bin directory and run:  
   Jmeter-server
* This will start the JMeter server mode, listening for instructions from the master.

1. **Ensure Network Connectivity**

* Ensure the master and slave machines are on the **same network** or have appropriate access configurations (like VPN).

1. **Modify jmeter.properties**

* On the **master machine**, edit the jmeter.properties file (in the bin directory) and set the following:  
   remote\_hosts=192.168.1.74
* If using multiple slaves, list them as a comma-separated value:  
   remote\_hosts=192.168.1.74,192.168.1.75

1. **Run the Test**

* Execute the command you provided:

jmeter -n -t D:\Jmxfiles\03\_10\_2024\_Update\All\All.jmx -l C:\apache-jmeter-5.6.3\bin\1160APIs1MinRamp5Distribute.csv -R 192.168.1.74

* The master sends the test plan to the slaves, executes it, and collects the results in the specified .csv file.

### **Generating Reports After the Test :- Once the test completes, you can generate an HTML report for detailed analysis:-**

jmeter -g C:\apache-jmeter-5.6.3\bin\1160APIs1MinRamp5Distribute.csv -o C:\apache-jmeter-5.6.3\bin\report\_output

* -g: Path to the results file (.csv file).

### **Key Points for Distributed Testing :-**

* Ensure all machines (master and slaves) have the **same JMeter version** and **test plan**.
* The jmeter-server process on the slave machines must remain active during the test.
* Use **firewalls or VPNs** to address connectivity issues between master and slaves.

Already mentioned this [All.jmx](https://drive.google.com/file/d/1MDT0x02FLbKvPdU_gSrZtzp1yrGRid3s/view?usp=drive_link) file here.   
[All.jmx](https://drive.google.com/file/d/1MDT0x02FLbKvPdU_gSrZtzp1yrGRid3s/view?usp=drive_link) file you can Download it from here.