**Questions in Apache JMeter**

1. What is the ideal duration period in load testing? How long will the load remain on the system when doing load testing?

***General recommendations:***

* ***Short-term load tests: For initial testing or identifying quick performance issues, 5-15 minutes might suffice.***
* ***Endurance tests: To assess system behavior under prolonged load, consider running tests for several hours or even days.***
* ***Peak load tests: Simulate expected peak traffic for a shorter duration (e.g., 30 minutes to an hour) to evaluate system performance under high stress.***

***The duration of the load in a load test depends on the specific goals of your testing.***

***Example:***

***Short-term Load Tests:***

* ***Duration: 5-15 minutes***
* ***Purpose: Identify quick performance issues or get a baseline understanding of system behavior.***

***Endurance Tests:***

* ***Duration: Several hours to days***
* ***Purpose: Assess system behavior under prolonged load, check for memory leaks, or other degradation issues.***

***Peak Load Tests:***

* ***Duration: 30 minutes to an hour***
* ***Purpose: Simulate expected peak traffic and evaluate system performance under high stress.***

Also what would be the ramp-up period and duration for the below scenario

Scenario: 400 users booking an appointment in the system per Day. The estimated duration a user can finish booking is around 5 minutes.

***Ramp-up period: 10-15 minutes***

***Load duration: 2000 minutes (or around 33 hours)***

***Total test duration: 30-60 minutes***

1. Is it heavy if multiple thread groups are used in a test script? Is it advisable to use multiple thread groups?

***Heavy? Yes.***

***Advisable? Yes, using multiple thread groups is often advisable.***

1. Embedded resources
   1. How do we handle the size of embedded resources since it consumes a lot of memory? Are there any adjustments/configurations in the test machine to accommodate embedded resource sizes? Or is it already the configuration of heap space?

* ***Use HTTP Cache Manager, exclude unnecessary resources, optimize HTTP Request Defaults, and consider test plan optimization.***
* ***Adjust heap space, optimize test machine***
  1. A failed request out of X number of embedded resources will already mark the main request as failed, is it safe to ignore embedded resources failure (like updating the configuration for the JMeter) so that the main request will not mark as failed? Will it not mask any possible issues in the server?

Ignoring embedded resource failures can mask potential server issues. While it might prevent main request failures, it's generally not recommended. Prioritize addressing embedded resource issues to get a complete picture of system performance.

1. What is the best practice when doing load testing with thousands of users?
2. Sample scenario: An estimated 8,000 users will be accessing the website per day, do we need to conduct load testing of 8,000 users or is it enough if we divide the 8,000 users per hour and then do the load testing for an hour?

***Simulate peak user load. For 8,000 users/day, focus on peak hours. Divide users accordingly and test for that duration. Consider additional load tests for different user patterns.***

1. When should we use the Transaction or Throughput controller? Currently, we only use a Simple controller to group elements, and a loop controller if needed based on the scenario.

***Transaction Controller: Group logically related requests.***

***Throughput Controller: Control the number of executions for specific samplers.***

* ***Use Transaction and/or Throughput Controller:  
  When you need to measure the end-to-end performance of a transaction or business process, rather than individual requests.***
* ***When you want to track the success/failure rate and response time of a group of related requests.***
* ***When you need to generate load that mimics real-world user behavior more accurately.***

1. Load Testing environment
   1. Is it better to conduct load testing through the LAN, through a WiFi connection, or both?

***Recommended approach is to conduct load testing primarily through a LAN connection, and then supplement it with some testing through a WiFi connection. You can then compare it with live public internet actual testing.***

* 1. Is it better to conduct load testing using a single high-end machine or multiple machines doing distributed testing? Will the Garbage Collector take longer? A single high-end machine can make faster loops and send more requests than a normal machine.

Single High-End Machine:

Advantages:

* Simpler to set up and manage compared to a distributed environment.
* May be more cost-effective if you have access to a powerful enough machine.

Disadvantages:

* The machine's resources (CPU, memory, network) may become a bottleneck, limiting the maximum achievable load.
* If the machine experiences issues (e.g., hardware failure, OS problems), the entire test is affected.

Distributed Testing:

Advantages:

* Can generate higher loads by distributing the workload across multiple machines.
* Reduces the risk of a single point of failure, as issues with one machine won't bring down the entire test.
* Allows for simulating more realistic user scenarios by having users distributed across multiple locations.

Disadvantages:

* More complex to set up and manage compared to a single machine.
* May require additional investment in hardware and infrastructure.

Regarding the Garbage Collector (GC) performance, the impact may vary depending on the specific scenario:

* In a single high-end machine scenario, the GC may take longer to clean up memory, as there is a higher memory usage and more object allocation happening on a single machine.
* In a distributed testing scenario, the GC load may be spread across multiple machines, potentially reducing the GC pause times on each individual machine.

1. What other performance monitoring tools are to be considered aside from the SSHMon JMeter plugin and HTOP?

***JMeter Plugins:***

* ***PerfMon Metrics Collector - Collects various system-level metrics (CPU, memory, disk I/O, network) from local and remote machines.***

***Built-in JMeter Listeners:***

* ***Aggregate Report - Provides summary statistics for the test, including response times, throughput, and error rates.***
* ***View Results Tree - Displays the response data, including response times and status codes, for each sampler.***
* ***Graphic Results - Generates graphs and charts to visualize the test results.***

***Third-Party Monitoring Tools:***

* ***New Relic - Provides comprehensive application performance monitoring, including integration with JMeter.***
* ***Dynatrace - Offers full-stack monitoring and analysis, with the ability to integrate with JMeter.***
* ***Datadog - Collects and analyzes system-level metrics, with the option to monitor JMeter tests.***

1. What is the recommendation when generating huge dynamic test data? Prepare test data manually in a CSV file or generate dynamic test data using JMeter?

For the preparation of test data in a CSV file, is it possible to use the CSV file in a distributed testing environment?

* For huge datasets, dynamic generation is more efficient and flexible than manual CSV creation.
* Yes, CSV files can be used in a distributed testing environment. Ensure the CSV file is accessible to all JMeter instances.

1. When does the ramp up period stop increasing when the user increases and the ramp up also increases? Given that the ramp up period increases as the number of users increases, how can we determine, for instance, that the system cannot handle X number of users?

The ramp-up period will stop increasing once the target number of users is reached, but you can determine the system's limitations by gradually increasing the user load and monitoring the system's performance and resource utilization. The point at which the system starts to degrade significantly is the maximum number of users it can handle.

1. JMeter generated report does not show response body messages which may contains detailed errors (like an SQL error of “Too many connections”) that may be helpful when resolving issues, but is visible in the View Results Tree when running via GUI. Is it still possible to get the response body or get such errors when doing load testing thru command line?

Yes, it is possible to get the response body and detailed error messages when running JMeter load tests through the command line.

Use the -l or --logfile option:

* you can use the -l or --logfile option to save the test results to a log file.
* This log file will include the response body and any error messages that were generated during the test.
* Example command: jmeter -n -t test.jmx -l results.jtl

Use the -e or --jtl-output option:

* The -e or --jtl-output option allows you to specify the output file for the test results in CSV or XML format.
* These output files will include the response body and error messages, in addition to the performance metrics.
* Example command: jmeter -n -t test.jmx -e -o results/

Use the -j or --jmeter-log option:

* The -j or --jmeter-log option allows you to specify the JMeter log file, which can provide additional information about the test execution, including any errors or exceptions that occurred.
* This can be useful for troubleshooting issues that may not be visible in the main test results.
* Example command: jmeter -n -t test.jmx -j jmeter.log

1. JMeter error interpretations, is it an error on the JMeter side or the target server?
2. Non-HTTP response code: java.net.SocketTimeoutException/Non HTTP response message: Read timed out

* This error indicates that JMeter is not receiving a response from the target server within the specified timeout period.
* The most likely cause is that the target server is taking too long to respond, or the network connection between JMeter and the target server is slow or unstable.
* It may also be caused by the test scenario being too aggressive, with too many concurrent users or requests.

1. Non HTTP response code: org.apache.http.conn.ConnectTimeoutException/Non HTTP response message: Connect to xxx.psa.gov.ph:443 [xxx.psa.gov.ph/192.168.xx.xxx] failed: Connect timed out

* This error indicates that JMeter is unable to establish a connection to the target server within the specified timeout period.
* The most likely cause is that the target server is not responding or is not accessible from the JMeter machine.
* This is likely an issue on the network or target server side, such as firewall rules, network configuration, or server availability problems.
* It may also be caused by the target server being overloaded and unable to accept new connections.

1. Interpretation of Reports
   1. Should we look and get the Average response time per API OR should we get the Total Average Response time of the APIs as a whole?

How about the 90, 99 percentile? How should we also consider or focus on those values?

Average Response Time per API:

* Analyzing the average response time for each individual API can provide valuable insights into the performance of specific parts of your application.
* This information can help you identify any APIs that are performing poorly or taking longer than expected, allowing you to focus your optimization efforts.
* Monitoring the individual API response times can also help you understand the impact of changes or updates to specific components of your application.

Total Average Response Time:

* The total average response time across all the APIs provides an overall indication of the application's performance from the end-user's perspective.
* This metric gives you a high-level view of the application's responsiveness and can help you understand the cumulative impact of the individual APIs.
* The total average response time is particularly important when evaluating the user experience and the application's ability to meet performance SLAs (Service Level Agreements).

90th and 99th Percentile:

* The 90th and 99th percentile response times provide insights into the performance outliers and the worst-case scenarios.
* The 90th percentile represents the response time at which 90% of the requests are faster, and the 99th percentile represents the response time at which 99% of the requests are faster.
* These metrics are crucial for understanding the overall user experience, as they help identify any instances of unacceptably slow response times, even if they occur infrequently.
* Monitoring the 90th and 99th percentile response times can help you set appropriate performance thresholds and ensure that your application can consistently meet the expected performance levels.

1. How can we simulate realistic user behavior with JMeter? (Think Time, Pacing, Randomization)
   1. the pauses or delays a real user would take between actions
   2. the rate at which JMeter sends requests within a Thread Group
   3. randomizing data inputs, think times, or navigation paths

Think Time:

* Think time, also known as "pause time" or "delay time," refers to the pauses or delays a real user would take between actions.
* In JMeter, you can use the Thread Group and the Constant Timer, Gaussian Random Timer, or Uniform Random Timer elements to introduce think time between requests.
* This helps mimic the natural pauses that users take while reading, thinking, or performing other actions.

Pacing:

* Pacing refers to the rate at which JMeter sends requests within a Thread Group.
* By default, JMeter sends requests as fast as possible, which may not reflect real-world user behavior.
* To simulate a more realistic pacing, you can use the Constant Throughput Timer or Throughput Shaping Timer elements.
* These timers allow you to control the rate at which requests are sent, ensuring a more gradual and consistent load on the system.

Randomization:

* Randomization helps introduce variability in your test scenarios, mimicking the unpredictable nature of real-user interactions.
* You can randomize the following aspects:
  + Data inputs: Use the CSV Data Set Config or Random Variable elements to randomize request parameters and data.
  + Think times: Combine the Gaussian Random Timer or Uniform Random Timer with the Thread Group to introduce random think times.
  + Navigation paths: Use the If Controller or Switch Controller elements to randomly select different navigation paths within your test scenario.