UnCodemy

JMeter Performance Testing Notes PDF

Apache JMeter is an open-source tool for load and performance testing of web applications and APIs. A JMeter test plan has thread groups that simulate users, samplers that send requests, config elements and timers, assertions and listeners. These notes explain each part, how to parameterise data, run tests from the command line and read the main metrics.

Key takeaways

- A test plan is built from thread groups, samplers, config elements, timers, assertions and listeners.
- Run real load tests in non-GUI mode; use the GUI only to design and debug.
- Response time percentiles, throughput and error rate are the key results.
- Add think time and realistic data so that the load resembles real use.
- Test only systems you are allowed to test, in an environment agreed with the owner.

What is JMeter performance testing?

Apache JMeter is an open-source Java application that generates load against servers, web applications and APIs to measure how they perform, using virtual users defined in a test plan.

What does JMeter do?

JMeter simulates many users sending requests to an application and records how the system responds. Teams use it to check response times under load, find the point where performance falls and confirm that fixes help. It works with HTTP and HTTPS, REST and SOAP APIs, databases through JDBC and other protocols. It does not run a browser, so it does not measure browser rendering time.

What are the parts of a test plan?

Element	Purpose
Test Plan	The container for everything
Thread Group	Defines the number of virtual users, ramp-up and loops
Sampler	A request, such as an HTTP Request
Config Element	Defaults and data, such as HTTP defaults or CSV data
Logic Controller	Controls order and conditions

Element	Purpose
Timer	Adds delay between requests
Pre/Post Processor	Extracts or prepares values
Assertion	Checks that a response is correct
Listener	Shows or saves results

How do I set up a thread group?

Number of threads is the count of virtual users. **Ramp-up period** is the time to start all users, so 50 users with a 50 second ramp-up start one per second. **Loop count** is how many times each user repeats the scenario, or use a duration for time-based tests. Start small, then increase the load in steps.

How do I create a basic test?

1. Add a Thread Group. 2. Add **HTTP Request Defaults** with the server name and protocol. 3. Add an **HTTP Request** sampler with the path and method. 4. Add a **Response Assertion** to check the status code or text. 5. Add a **View Results Tree** while debugging and a **Summary Report** or a file-based listener for runs. 6. Run with a few users and check that each request is correct before scaling.

How do I make the test realistic?

Add **timers** (such as a Constant or Gaussian Random Timer) to mimic think time between actions. Use a **CSV Data Set Config** so that each user logs in with different test data. Use a **Cookie Manager** and a **Cache Manager** where a browser would. Model the real mix of actions, for example many searches and fewer purchases, using throughput controllers.

```
username,password
testuser01,TestPass#01
testuser02,TestPass#02
```

Use made-up test accounts, and never put real user passwords in test files.

How do I handle dynamic values?

Many apps return tokens or IDs that you must send in the next request. Use a **JSON Extractor** or **Regular Expression Extractor** to save the value in a variable, then use it as `${token}` in later requests. This is called correlation.

How do I verify responses?

A **Response Assertion** checks the status code or content. A **JSON Assertion** checks a value in JSON. A **Duration Assertion** fails a request slower than a limit. Use assertions in moderation under heavy load, since each one costs CPU on the test machine.

How do I run a real test?

Use the GUI to create and debug, then run in non-GUI mode, which uses fewer resources:

```
jmeter -n -t plan.jmx -l results.jtl -e -o report/
```

`-n` is non-GUI, `-t` the plan, `-l` the results file and `-e -o` generates the HTML dashboard. For large loads, use several load generator machines in distributed mode so that the test machine is not the bottleneck.

Which results matter?

Metric	Meaning
Average and percentiles (90th, 95th, 99th)	Response time; percentiles show the slow tail
Throughput	Requests handled per second
Error %	Share of failed requests
Active threads over time	Load applied
Connect and latency times	Network versus server time

Do not rely on averages alone, because a few very slow requests can hide behind a good average. Compare results with agreed targets, for example "95 percent of search requests under two seconds", and monitor the server's CPU, memory and database during the test.

What types of tests should I run?

Load (expected users), **stress** (beyond capacity to find the limit), **spike** (sudden jumps) and **soak** or **endurance** (long periods to find leaks). Run one type per test and change one variable at a time.

What are the common mistakes?

Running heavy tests in the GUI, no think time, testing with one user account, not warming up, ignoring the load generator's own limits, testing a production system without permission and judging results from one run. Repeat runs to check consistency.

What should I practise?

Install JMeter, test a public demo site that allows load testing or your own local app, build a plan with ten users, add an assertion and a timer, run it in non-GUI mode and open the HTML report. Then double the users and compare the 95th percentile response time. Writing two lines about what changed teaches you more than reading the metrics list.

Frequently asked questions

What is JMeter used for?

JMeter is an open-source tool that simulates many users to test how web applications, APIs and databases perform under load.

What is ramp-up in JMeter?

It is the time JMeter takes to start all virtual users, so 50 users with a 50 second ramp-up start about one user per second.

Why use non-GUI mode?

Non-GUI mode uses fewer resources and is more reliable for real load runs, while the GUI is for designing and debugging.

What is correlation in JMeter?

Correlation means extracting a dynamic value, such as a token, from one response and using it in later requests.

Which results should I look at?

Response time percentiles, throughput and error rate, together with server CPU, memory and database metrics.

Sources and further reading

- Apache JMeter user manual
- Apache JMeter

About this PDF

These PDFs are study material for learning and revision. They are not a certificate, a degree or a promise of a job, and technology changes, so check official documentation for the latest details. Spot an error? Write to info@uncodemy.com and the trainer team will review it.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550