

Scalability Factors of JMeter In Performance Testing Projects



Title	Scalability Factors for JMeter In Performance Testing Projects
Conference	STEP-IN Conference – Performance Testing 2008, PUNE
Author(s)	Budhaditya Das (Aztecsoft iTest, Pune) Priyanka Mane (Aztecsoft iTest, Pune)
E-mail ID(s)	budhaditya.das@aztecsoft.com priyanka.mane@aztecsoft.com
Address:	Aztecsoft Ltd. Rajiv Gandhi Infotech & Biotech Park, Plot no. 37, Phase 1, MIDC, Hinjewadi, Pune – INDIA 411057

INTRODUCTION

Testing web applications for performance is fundamentally different and complex than testing them for functional correctness. Various flavors of load generation tools are available to simulate the expected load levels on servers, network or web application to test their ability to sustain concurrent heavy load under realistic conditions. In order to carry out effective performance testing of web applications one has to ensure that sufficiently powerful hardware is used to generate required load levels. At the same time, one would prefer to avoid investing in unnecessarily expensive hardware “just to be sure”. Having an effective model for estimating the load generation capabilities of load generation tools on different hardware configurations can greatly help in taking care of both these requirements.

Unfortunately, the only such model we currently have is the simplistic set of benchmarks offered by different Commercial Performance test tool vendors. These benchmarks spell out how many “virtual users” their tool can simulate on different hardware configurations. In fact the load generation capability of any tool is a function of multiple factors and not just the underlying hardware configuration. Hence, the simplistic model is therefore not usable in practice. Some of the other factors that affect the load generation capacity of a tool include application response sizes, application response times and the complexity of client-side activity.

The Performance Engineering group from Aztecsoft’s Expert Services Group attempted to create such a model for Performance testing tool “JMeter”. We have created an experimental setup wherein we measured the sensitivity of JMeter performance to some of the factors described above and used this data to construct the model. This paper describes our experimental setup and presents our model for determining load generation capabilities of JMeter.

MOTIVATION BEHIND SCALABILITY EXPERIMENTS

During the course of our work on web application performance testing we have had occasions to evaluate and use a number of commercial as well as open source load generation tools. But independent of the tool in use, some questions that invariably pop up during various phases of Performance testing life cycle are:

- **“How many load generators”** would be required to generate “X” no of virtual users ^[1]?
- What is the **“configuration of load generators”** required to simulate the expected number of virtual users? Or conversely, for a given commodity hardware what is the maximum number of virtual users that can be simulated efficiently?

In an attempt to answer these frequently asked questions, most commercial tool vendors publish standard benchmark figures highlighting the load generation capability of their tools. Typically the load generation capability chart for any commercial load generation tool would be represented as below:

Hardware Configuration	Max # Virtual Users per Agent
4xPIII, 1GHz, 4 GB RAM	5900
2xPIII, 1GHz, 2 GB RAM	3600
1xPIII, 1GHz, 1 GB RAM	1900
1xPIII, 1GHz, 512 MB RAM	800

Table 1: Load generation capability chart for a popular commercial load generation tool

The above table indicates that the load generation capability of the tool (in terms of # virtual users per machine/agent) is a function of the underlying hardware configuration. While it is true that hardware configuration plays an important role in determining the load generation capability of the tool, it is also true that this is not the *only* deciding factor.

In fact, the “# Virtual Users per machine” that can be supported by any load generating tool is not only a ‘function’ of the underlying hardware but also depends on various application (application under test) specific parameters and tool configurations. It therefore becomes crucial to examine these factors affecting the load generation capability of the tool.

¹ A virtual user is an emulation of a real-world user, whose actions are described in a test script that is compiled before test execution. In other words Virtual users are scripts that emulate the steps of a real user using the application. To interact with a remote software system a virtual user either communicates directly via a certain communication protocol (for example, HTTP/HTTPS, LDAP)

1 FACTORS AFFECTING LOAD GENERATION CAPABILITY OF TOOLS

A quick look at the design of load generation tools would provide meaningful insight into the factors that affect tool performance.

1.1 How Load Generation Tools Work

Load generation tools are designed to simulate the interaction of the “real” users with the application under test. These tools enable us to test application performance by spawning a large number of virtual users from a single load generating machine.

At the heart of a load generation tool is the core engine that works at the protocol level to generate the traffic/requests which would normally be generated by “real” users driving the user interface of the application under test. These requests form the basis of a load test script and in essence this is what is replayed by the ‘Virtual Users’ against the application/system under test.

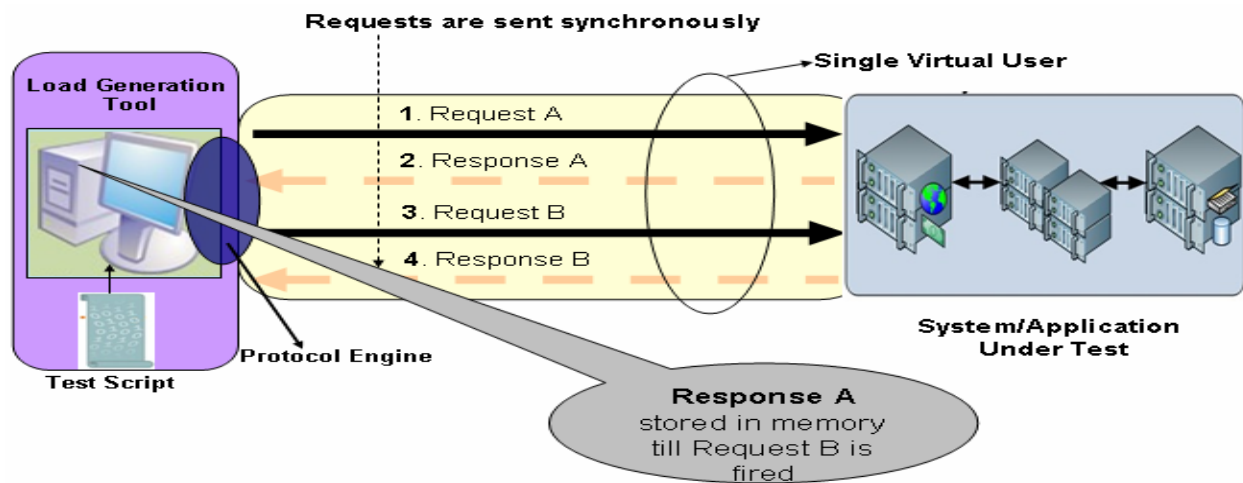


Figure 1: A simplified view of “How Load Generation Tools Work!”

Here is how the load generation tool fires a set of requests for a single virtual user:

It is important to note that the protocol engine fires requests synchronously

- Protocol engine fires *Request A* to application under test
- Protocol engine waits for *Response A* before it proceeds with execution of *Request B*
- Once Response A is received by the protocol engine, it is stored in memory for analysis and processing. This response is discarded from memory only after Request B is sent

Every simulated virtual user has a “cost” associated with it in terms of CPU and Memory “footprint”. Hence the maximum number of virtual user that can be simulated on a given hardware is dictated by the

average memory/CPU footprint for each virtual user. The memory/CPU footprint is in turn affected by application response and the complexity of the script to be simulated.

1.2 Scalability Factors

Every web application is different; its response time and size vary each time. As a result of this, the actual number of virtual users/ load levels the tool can simulate for each application is different. Along with this, the overhead for simulating each protocol (e.g. HTTP vs. HTTPS) is different. Hence the maximum number of virtual users is also affected by the underlying protocol.

Load test script complexity also limits the load generation capability of a tool. A complex script (i.e. more regular expressions parsing, conditional statements etc.) results in lower number of virtual per machine. Finally, the load generation tool configurations also affect the performance of the tool. Examples of these tool specific configurations include logging level, output format, (J)VM parameters and GUI/ non GUI mode of execution.

The various factors that affect the scalability of any load generation tool can be categorized as follows:

- **Application specific factors**
 - Average size of response
 - Average response time
 - Underlying protocol
- **Load generation tool related**
 - Complexity of client-side processing
 - Load generating tool architecture and configuration etc.
- **Hardware configuration of the load client (machine hosting the load generation tool)**

2 JMETER SCALABILITY EXPERIMENTS

The load generation capability of a tool is sensitive to the factors mentioned above. We therefore designed and carried out a set of tests, in a controlled environment, in an attempt to experimentally establish a correlation between the optimal # of virtual users (that the tool can generate) across the various aforementioned dimensions/factors. The experiments are designed around JMeter, a versatile, open source load generation tool that we had already used across various load testing engagements.

These set of expansive experimental tests have been christened as the Scalability Experiment. The scalability experiment attempts to answer questions like:

- *What load levels can we achieve using commodity hardware?*
- *What is the effect of application "Response Time" on load generation capabilities of JMeter?*
- *How does large "Response Size" degrade load generation capability of JMeter?*
- *What is the effect of the underlying protocol on load generation capability of JMeter?*
- *How does JMeter capability vary with nature of script contents?*
- *How does tool configuration (say for example log type csv v/s xml) affect JMeter performance?*

As we know that the number of virtual user per machine is sensitive to the above mentioned parameters and we therefore attempted to establish a correlation between the optimal number of users and each of the parameter. **In our theoretical model we defined the "optimal number" as that number of virtual users beyond which increasing the number of virtual users does not lead to an increase in load throughput even when the web server and network have not been stretched to their limits.** Implicit in this model is the assumption that the load generating capability of a client increases with increasing number of virtual users until this optimal number of virtual users is reached and that there is degradation of performance thereafter.

While measuring the capabilities of performance test tool it is important to ensure that factors like the test application itself or network of the test environment do not become a bottleneck. To achieve this, the test application needs to be designed in such a way that it can be meaningfully controlled and guided by the performance tester. In other words, the above prerequisite states that the application response time and data size should be configurable parameters of the application. Hence we designed a test application called 'JScaleAppLite', which is designed to return data of custom data size. The test application allows the expected data size and response time to be specified as part of the URL parameters (http://PerfTestApps/JScaleAppLite/index.aspx?data_size=20480&resp_time=1000)

Along with the test application, it is crucial to avoid bottlenecks in network of the test environment. In order to ensure this we designed a separate subnet to execute scalability tests. Not only the application and network are the possible sources of bottleneck, but also application deployment is another potential area. Predicting that single deployment could be overwhelmed by application requests we used multi-deployment environment, for distribution of requests avoiding application/web server bottleneck issues.

3 SCALABILITY EXPERIMENTS RESULTS

The results prove the stated sensitivity of load generating capability of JMeter version 2.1.1 with aforementioned scalability factors. The following graphs are representative of the effects of the same. The current set of tests does not take into consideration the complexity of the script (i.e. the client side processing). All the scripts are of simple^[2] complexity.

3.1 Effect of Response Time

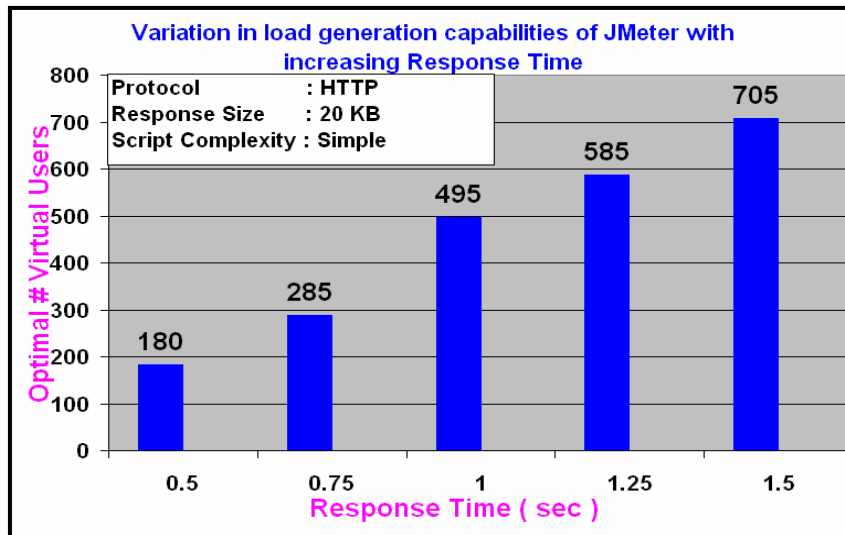


Figure 2: Effect of application "Response Time"

The above graph depicts the variation of optimal number of virtual users for various response time values for a constant response size of 20 kb. **The optimal number of virtual users increases with increase in response time.** It increases from around 180 virtual users to around 700 optimal virtual users when the response time changes from 500 ms to 1.5 seconds. This is approximately a 380% increase (for a constant response size of 20 kb over HTTP protocol, nature of script complexity is simple).

² A simple script indicates minimal client side processing in the form of response parsing or dynamic request generation. The simple script includes a single request without intermediate conditional analysis , regular expression based response parsing or dynamic request creation

3.2 Effect of Application Response Size

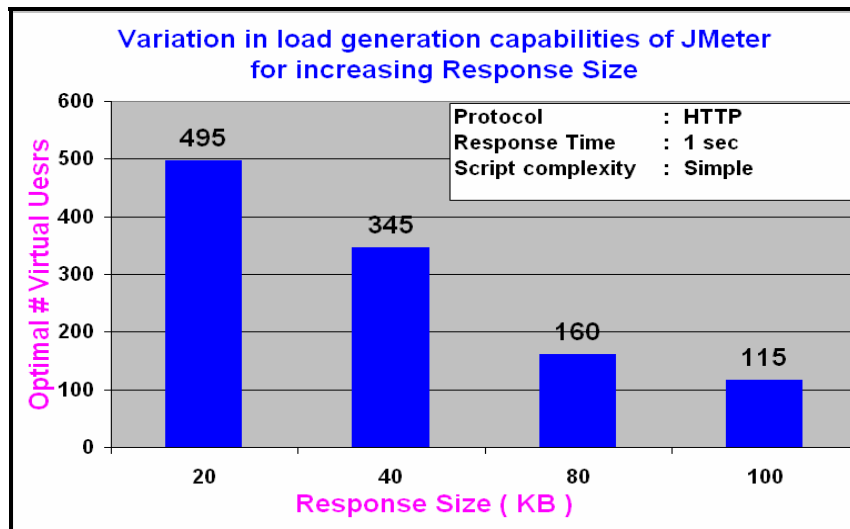


Figure 3: Effect of application “Response Size”

The above graph depicts the variation of optimal number of virtual users for various response size values (20 kb, 40 kb, 80 kb, and 100 kb). Application response size has a massive effect on the load generation capabilities of JMeter. The optimal # of virtual users drops from around 500 virtual users to a measly 115 virtual users when the application response size increases from 20 kb to 100kb. This is approximately a 350% drop in the optimal number of virtual users (for a constant response time of 1 second over HTTP protocol, nature of script is simple).

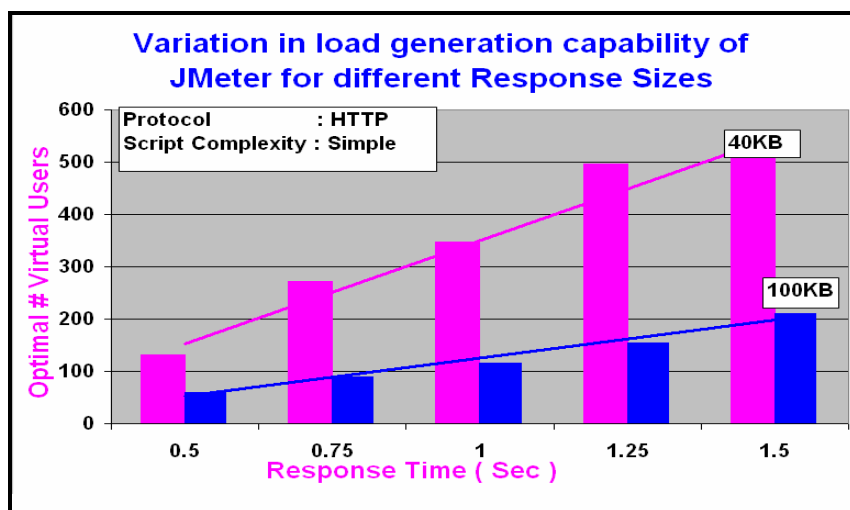


Figure 4: Variation in load generation capability of JMeter across response sizes

The above graph depicts a comparative representation of the difference in the load generation capability of JMeter for response size of 20 KB and 40 KB for increasing response times (between 0.5 sec and 1.5 sec, protocol –http and script complexity - simple)

3.3 Effect of protocol change

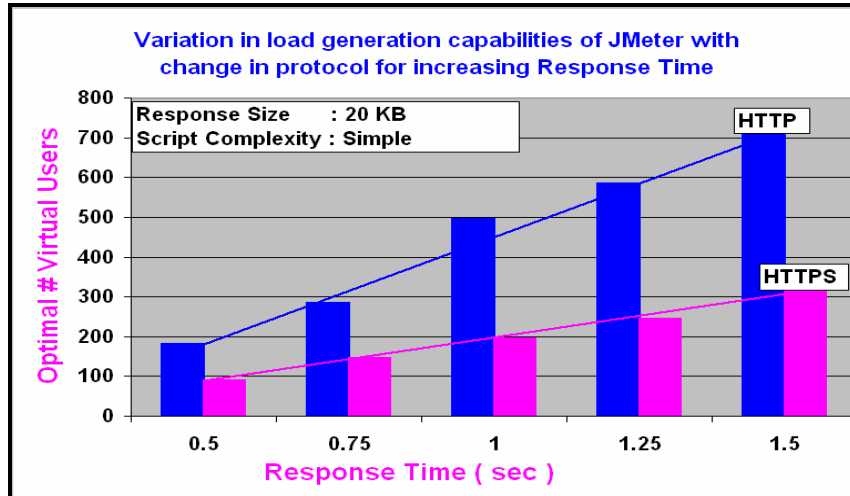


Figure 5: Effect of underlying "Protocol"

Considerable decrease in load generation capability of JMeter is observed when the underlying protocol changes from http protocol to https protocol. This is represented by graphs depicted above. **The load generation capability decreases by 50% or more when the protocol is HTTPS** (script complexity is simple)

4 AZTECSOFT JMETER SCALABILITY MATRIX

The scalability experiments have been carried on commodity hardware with the following configuration:

Hardware Configuration of load generator	Load Tool Configuration
OS : Windows XP/SP2	Load generation tool: JMeter
Processor : Intel Pentium 4 2.4 GHz	Tool version : 2.1.1
Memory : 2GB RAM	JDK version : 1.5.0
Disk space : 40 GB HDD	

The outcome of the scalability experiments has been tabulated in the form of “scalability matrix”. Given the application specific parameters (protocol, response size, and response time) as input, this matrix can be used to identify the optimal number of virtual users that can be simulated on a single load generator (with commodity hardware). The findings of the scalability experiments are summarized below in form of scalability matrix (partial) given below:

Protocol	Response Size(KB)	Response Time(ms)	Optimal # VUsers	Optimal Throughput (requests/sec)
HTTP	20	500	180	213
HTTP	20	750	285	243
HTTP	20	1000	495	265
.
HTTP	80	750	115	132
HTTP	80	1000	160	135
HTTP	80	1250	220	136
.
HTTPS	5	1250	325	215
HTTPS	5	1500	395	229
HTTPS	10	750	190	216
.
HTTPS	20	1500	315	177
HTTPS	40	500	60	107

Table 2: JMeter Scalability Matrix (Partial...)

5 PRACTICAL APPLICATION OF SCALABILITY TEST RESULTS

The scalability test results have a direct application in various phases of performance testing life cycle like test strategy and planning, test execution and analysis phase.

Estimating the number of load generators required to simulate the expected load levels is an important activity in performance test strategy and planning phase. Here the # of load generators required can be calculated as:

$$\frac{\text{Max \# of virtual users that can be simulated per load generator } m/c}{\text{Total virtual users that need to be simulated}}$$

With the scalability matrix as reference, the Performance analyst would simply need to identify a few application specific parameters (application response time, application response size, protocol of application) and map this with the scalability matrix in order to accurately estimate the required number of load generators.

As per the central thesis of scalability experiments, the load level handled by load generator is sensitive to minimum application response time and maximum application response size, protocol of that application and nature of processing. Dividing the expected load levels by the capability of single load generator obtained from scalability results would determine the number of load generators. This figure helps to determine the load generation hardware of performance testing lab.

Once a performance test run is complete it is required to validate the test results to identify the potential performance bottlenecks. For tests with high load levels, it is important to make sure that the load generation tool has not become a bottleneck. Without scalability results at hand this would be an ad hoc process of executing tests until it is identified that load generation tool has become a bottleneck. The scalability test results help in establishing the upper limit for load generation capability of the tool. Having known this threshold value, the performance tests can be designed such that time spent in executing tests is optimized.

6 THE ROAD AHEAD...

We have covered only a few aspects in the scalability experimentation of JMeter so far. To complete the experimentation from all the angles, we plan to carry out these experiments involving other factors like:

- Script complexity
- Tool configuration

7 REFERENCES

- <http://jakarta.apache.org/jmeter/> .
- *The official site for the load testing tool "Jakarta JMeter".*
- http://mail-archives.apache.org/mod_mbox/jakarta-JMeter-user/200206.mbox/%3C3D08D062.19403.6A23FE5@localhost%3E . *A mail thread from JMeter forum with pointers to the fact that the inherent JMeter distributed mode doesn't scale up effectively*
- http://www.segure.com/pdf/silkperformer_reviewers_guide.pdf . *Silk performer user guide. Page 21 tabulates a set benchmarks describing the load generation capabilities of the tool on different hardware configurations*

8 AUTHOR PROFILES

This paper is co-authored by Priyanka Mane and Budhaditya Das from the iTest Practice group of Aztecsoft.

Budhaditya Das is presently working as a Test Analyst at Aztecsoft iTest Expert Services Group (Performance Testing). As a member of this group he is responsible for R&D, tool development, pre-sales and training in Performance Testing. In addition his areas of interest also include Web Application Security testing and test automation. He has Bachelor's degree in Electronics from Shivaji University, India. He has 3.5 years of experience in the software industry.

Priyanka Mane is presently working at Aztecsoft iTest and is responsible for R&D activities in the area of Web Application Performance Testing. She has Bachelor's degree in Computer Science and Engineering from Shivaji University, India.