International Journal of Engineering in Computer Science

E-ISSN: 2663-3590
P-ISSN: 2663-3582
IJECS 2019; 1(1): 57-62
Received: 20-11-2018
Accepted: 25-12-2018

Manideep Yenugula
Kohls, Milpitas, California, 95035, USA

Raghunath Kodam
Apple, California, 95035, USA

David He
Apple, California, 95035, USA

Correspondence
Manideep Yenugula
Kohls, Milpitas, California, 95035, USA

Performance and load testing: Tools and challenges

Manideep Yenugula, Raghunath Kodam and David He

DOI: https://doi.org/10.33545/26633582.2019.v1.i1a.102

Abstract
Applications built for mobile devices, websites, online services, the cloud, and grid computing may all undergo performance testing. Tool installation, tool application flexibility, tool configuration, response time created by the tool, etc. are all potential tool-related difficulties that may arise during performance testing. This study will examine the application's performance using two separate tools, one of which is SoapUI and the other is Apache Jmeter. Since many tools provide varied results, we need to choose the testing instrument that is most suited to the job at hand. As an alternative to SoapUI's demand testing tool, Apache-Jmeter is a web application performance assessment tool that helps examine the server's efficiency under severe stress. It evaluates an application's Quality of Services under variable demand.

Keywords: Performance testing, load testing, jmeter, soap UI

Introduction

Applications built for mobile devices, websites, online services, the cloud, and grid computing may all undergo performance testing. Problems with tools might develop during performance testing for a variety of reasons, including installation, application flexibility, configuration, response time, etc. [1]. Workload requirements are essential for conducting load testing using model-based performance prediction to evaluate application system performance features. There is a significant issue in both domains when it comes to correctly defining workload parameters that reflect the actual workload [2]. In order to construct new cloud-based distributed storage systems, Internet-based technologies nowadays are progressively shifting towards a service-oriented functionality, which brings with it great processing power, capacity, and flexibility. The smooth launch of cutting-edge online and social services like Twitter, Facebook, ring ID, etc. depends on a number of distributed systems that are active in many data centers throughout the world. Efficient capacity allocation, correct setup, and tuning of various system resources are essential for the proper functioning and reliability of both the cache as well as backend servers in distributed systems, ensuring that they can handle incoming workloads [3].

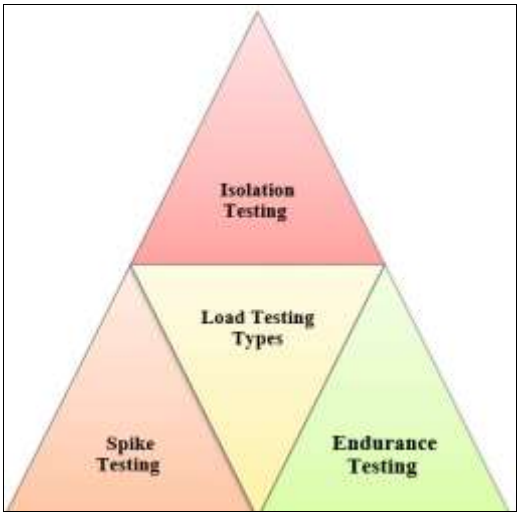

Fig 1: Load Testing Types

Tests are an integral part of the software development life cycle. When it comes to product quality, testing is king. The importance of ensuring the quality of a website or mobile application has grown since more firms increasingly focus on online applications. All the way through a software's life cycle, testing is conducted. Basic application functioning, graphical user interface, availability, performance under high traffic, etc. are all part of the web testing process. When it comes to testing, web-based apps present a number of obstacles. The software industry has created a plethora of tools and strategies to help reduce the difficulty of testing [4]. An application's scalability may be defined as their capacities to handle an increasing processing load as needed. It also discusses how to create a system on a smaller scale. Among the many quality attributes-performance, reliability, usability, accessibility, and security-testing scalability is among the most difficult. Nevertheless, empirically speaking, this area of study is under-researched [5]. An evaluation and successful demonstration of a new morphing flap system, developed in cooperation with the University of Bristol under the INNWIND.eu project, was carried out during a testing session that made use of DTU's outdoor rotating rig. Furthermore, ECN's recently developed aerofoil has been tested in real-world air conditions to assess its aerodynamic performance. Aerodynamic lift control is an area where the morphing wing excels, and its results are in line with those predicted by computational fluid dynamics [6]. In smart cities, the Norwegian Research Center for Zero Emission neighborhoods will define zero emission neighborhoods and test it with nine trial sites. Greenhouse gas emissions, power/load, energy, accessibility, economics, spatial characteristics, and innovation are the seven domains that the ZEN definition examines via a set of evaluation criteria and KPIs [7].

Related Work

The authors of the aforementioned study [8] provide a way to automate the extraction and translation of workload requirements, which will help with load testing and model-based performance forecasts of session-based application systems. The technique is composed on three main elements, which are WESSBAS. First, a system-and tool-agnostic domain-dependent language may be used to layered-model session-based system workload needs. Secondly, production system session logs automatically include instances of that DSL. The next step is to convert these examples into load generation tool and model-based performance assessment tool executable workload specifications. The Palladio Component Architecture and the widely used load evaluation tool Apache JMeter have both been modified by us. The SPECjEnterprise2010 benchmark and the access logs from the 1998 World Cup serve as the standard by which their method is measured. Extracted workloads closely resemble measured workloads with regard to workload-specific attributes (Such as session durations and arrival rates) as well as performance metrics (Such as response times as well as CPU utilizations).

Authors will use two separate tools, Apache-Jmeter and SoapUI, to examine the application's performance in [9]. They should pick the testing tool based on the application as various tools provide different replies. One excellent tool for measuring server performance under strong demand is Apache-Jmeter, which is based on web applications. In contrast, the SoapUI Load Testing application checks an application's QoS under varying loads.

In [10], the authors lay forth a method for evaluating server workloads quantitatively and attributing performance factors. First, they show that prior work using load testers was inaccurate and that careful planning for the server's load tester is necessary to guarantee high-quality performance assessment. By analyzing the shortcomings of previous work, they were able to create Treadmill, an adaptable load tester system that eliminates these problems. After that, authors build analysis and measurement processes that can correctly assign performance aspects using Treadmill. They use quantile regression and statistically sound performance assessment, tweaking it to fit server system quirks. Their improved technique concludes with an evaluation of the effects of common server hardware characteristics on Facebook production workloads. They show that their assessment approach yields better findings, especially when it comes to capturing complex and counter-intuitive performance patterns, and they break down the implications of these attributes on request tail latency. They saw a 43% drop in 99th-percentile latency and a 93% drop in its variation by adjusting the hardware characteristics according to the attribution's recommendations.

In order to evaluate the efficiency of cache with backend servers, the authors of [11] suggest a test-driven automated design called'svLoad' for load testing. In that example, they used technologies like JMeter and Ansible as well as some custom utilities bash scripts to create test cases that take into account a variety of real-world circumstances, such as various kinds of protocols, identical or distinct URLs, load or no load, cache hit or miss, and so on. To ensure their method is effective, they run these test cases on a real private cloud programming environment using two open source projects: OpenStack Swift for the backend with Varnish for the cache server. By running load requests, they want to identify Varnish and Swift bottlenecks and use that information to fine-tune the system. According to their test results, they were able to achieve an 80% improvement in response time after tweaking the network infrastructure, Varnish, and Swift.

To find out how well a system responds, how fast it can handle a given workload, how reliable it is, and how scalable it is, researchers in [12] employ the LoadRunner testing program. The strategy was used during the performance testing stage of the web application for purchasing airline tickets online. They identified the system's flaw while dealing with large numbers of users, and they used the results of the load tests to inform their recommendations for improvement.

According to [13], the main goal of that research is to catalog all the current metrics and techniques used to evaluate a system's scalability. Not to mention learning about the difficulties encountered when carrying out these measuring procedures. A comprehensive mapping research was carried out to gather data about the identification of indicators and tools for assessing scalability. Their goal in using that strategy was to compile aggregated data from a wide variety of online sources.

Exhaust diffuser and last-stage design considerations for LP exhaust systems over a broad operating range are the subject of research in [14]. A 1/10th scale air testing rig was built to guarantee that the mathematical fluid dynamics instruments could faithfully replicate the machine's performance under part-load circumstances, which are described as very

turbulent flows entering the diffuser. To find out what occurs, they do a numerical parametric study after restating the blades of the final stage rotor. The shape of the diffuser was also changed. Rearranging the rotor blades is one way to modify the flow characteristics at the diffuser inlet, which in turn alter the quantity of leaving energy as well as the diffuser's ability to recover departing energy. The benefits of restaggering the blades of the rotor and expanding the diffuser are not mutually incompatible, suggesting that they may be explored separately. Lastly, by considering the diffuser's resizing with the restaggering of the final stage rotors, an excellent design shown that performance may be improved. According to that idea, a typical 1000 MW plant that operates largely at part-load would see a 1.5 percent boost in last-phase power generation.

An extensive dynamic building simulation software is used in a semi-virtual environment laboratory setup, which involves controlling a genuine heat pump from inside a controlled environment and connecting it to loads of a virtual building. In that arrangement, MPC (Model predictive control) techniques are created and evaluated [15]. Here in the lab, they try out several MPC techniques to see whether one reduces the building's thermal energy delivery, the heat pump's operating expenses, or the CO_2 emissions from the system. In terms of various indicators, including costs, convenience, carbon footprint, as well as energy flexibility, the results demonstrate that the MPC controller is capable of load-shifting through the timing of thermal energy storage charges. The control strategies are evaluated for their satisfactory performance. Additional helpful insights are provided by discussing the actual issues that were experienced during the installation using a genuine heat pump.

Load Testing Tools & Challenges

When evaluating the functionality of software, websites, apps, and associated systems, load testing is an important part of the process. This test does not really work, but it mimics how the system would respond if several people tried to access it at once. One of the last and most important forms of testing done before deployment is load testing, often called "volume testing." It simulates real-world use to ensure the web system is stable, functional, and performs as expected.

Among the many important features of the web systems that are discovered via load testing are the following:

Considerations such as:

- The application's total operational capacity, which includes the amount of multiple users that can be supported at once;
- The application's response time, throughput costs, as well as demands on resources under various user load levels;
- The application's infrastructure stability.

Prior to releasing any client/server internet or intranet application, load testing is an essential procedure. Everything from front-end apps like websites to back-end systems like the servers that power them falls under this category.

Although functional tests are essential in software development, they cannot forecast how well a product will work under different user involvement levels. Before releasing software or installing updates, load testing helps businesses find and correct key performance problems that other tests miss.

There are three main reasons why businesses should do load testing:

- With this program, we can:
- Evaluate its capabilities
- Make money, provide service, and safeguard our reputation
- Make sure the user interface is easy to use and effective

Finding bottlenecks, measuring response times for site activities, and improving future performance all need load testing. These objectives may, of course, be satisfied in reaction to a live site's activity, although only at the price of severe disturbance to consumers.

Businesses big and small may reap the benefits of load testing. Here are a few examples of real-world applications of load testing:

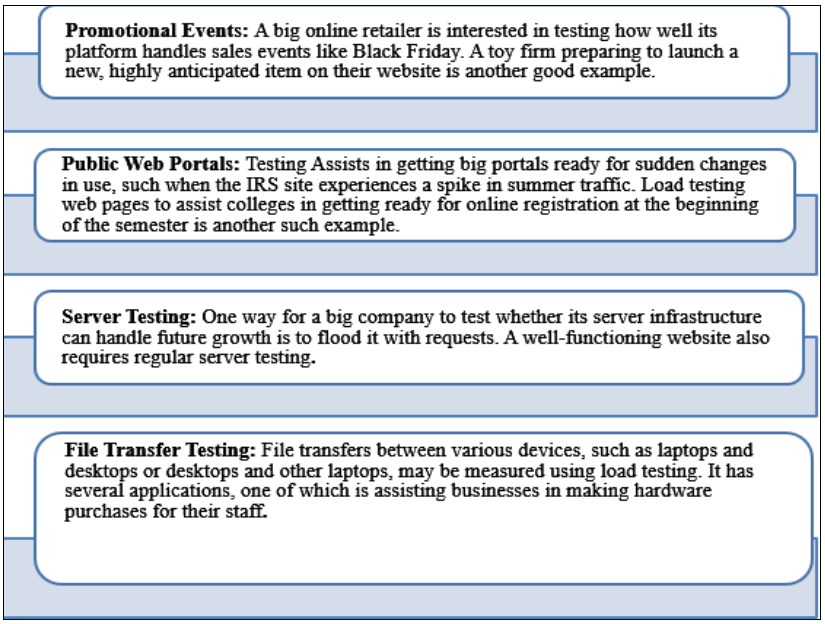

Fig 2: Test Cases / Applications of Load Testing

There are a lot of uses for load testing, including ones that most businesses don't think about. Just a few examples are:

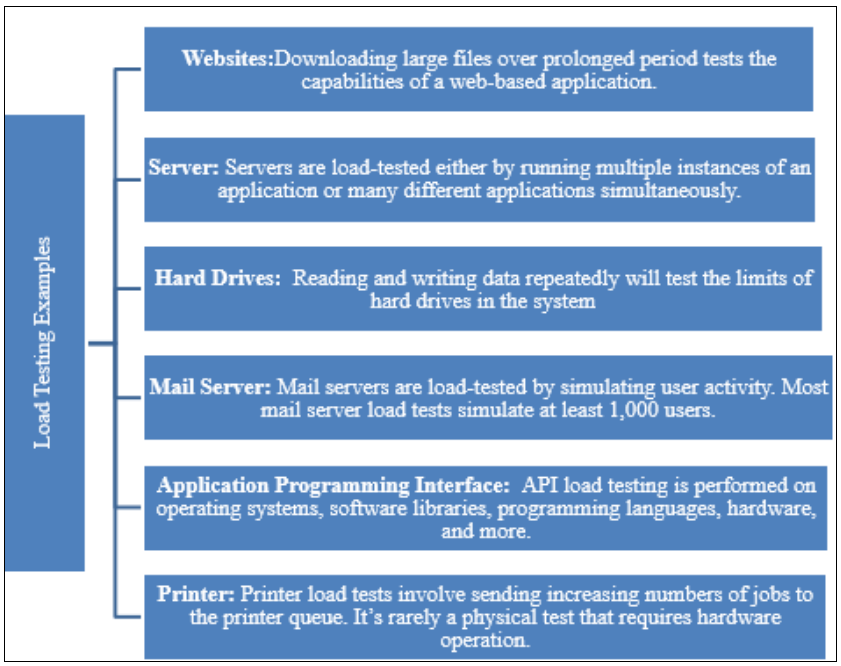

Fig 3: Load testing Examples

Challenges and Limitations of Load Testing

Load testing is widely used across many sectors and systems because of its many advantages. Nevertheless, there are disadvantages and difficulties, as with every application.

Challenge 1: Intangibility

Load testing isn't always the most eye-catching method, but it may help find issues before they happen in the real world. The financial and non-financial costs of site outage and application failure never come to fruition.

Types of testing that ask "what if" questions are often disregarded. Although load testing is useful for post-user-overload analysis, it is significantly more useful for organizations when used preventatively.

Challenge 2: Complexity

There may be a significant technological barrier to entry for both open-source as well as in-house load testing solutions. They might not have the manpower or funds to dedicate to load testing if their business is too small or too complicated.

Professional load testing platforms, like ZAPTEST load testing, will prioritize having an intuitive interface, thus this won't be a problem for them. In System under Load (SUL), ZAPTEST LOAD allows users to build API-based scripts that execute end-user company procedures and measure end-to-end transactions.

Load Testing Tools

Using open-source testing tools is often the first step for many businesses. Many choices are available, such as the following:

- JMeter – An enterprise tool called LoadRunner is the basis for this Java application.
- Taurus – A resource for creating one's own load testing.
- k6 – A load testing instrument designed for seasoned programmers with an emphasis on back-end architecture.
- SoapUI – The SoapUI load test makes use of the SOP. Furthermore, a paid edition of this program is on the market.
- Locust – A load-testing tool that is well-known for its low resource requirements and relative ease of use.
- LOAD Studio, a part of ZAPTEST FREE Edition, offers free performance testing, scripting via API, test recording, and functional testing correlation.
- Businesses should weigh the benefits and drawbacks of open-source testing tools before committing to them, even when they don't cost anything upfront.

Load test technologies provide several significant benefits.

Low Cost

For open-source software, the lack of cost is the primary benefit. It is possible to do load testing without investing any money, which is great for startups and other businesses with minimal resources.

Flexibility

The community regularly reviews, updates, and improves open-source software. Depending on your exact testing requirements, there may be available add-ons.

Faster Upgrades

Compared to commercial software, open-source software usually sees more rapid advancements. New features, security patches, and bug fixes often get out more frequently and with less disruption.

Limitations of Load Testing Tools

Companies should be aware of the possible downsides of free load testing tools, despite their great advantages.

Lack of Support

Users of open-source load software for testing are left to rely on community-based resources such as wikis and forums to resolve any difficulties that may arise. No one is

available by phone or email to help users of free tools, in contrast to commercial software.

Complexity
The focus of open-source load testing tools isn't always on making the product easy to use. The assumption that the user has some level of advanced programming expertise is made by many programs. It might be challenging to learn how to do load tests using open-source tools.

User Load Limitations
Problems with memory and CPU are common in open-source testing tools during high capacity load tests. Free load testing may not be robust enough for enterprise-level businesses.

Load Testing and Performance Testing Metrics
In order to learn more about scalability, dependability, and speed, the company runs non-functional simulations. When you test each of the aforementioned components separately, you get a fuller picture that makes it simpler to spot bottlenecks.

Baseline Performance
Businesses may check the application's baseline performance via load testing. Benchmark results for average link speed, file downloading time, and latency are shown by the data produced as the test's user count climbs progressively.

Benchmark Performance
You may also get statistics on benchmark performance using a website load test. Despite their frequent interchangeability, the terms "baseline" and "benchmark" really mean rather different things. Performance is evaluated by benchmark testing in relation to other websites or internal criteria, including end-user service level agreements.

Load Testing Metrics
Companies will tailor their testing measures to meet their own requirements. The capacity to tailor the metrics collected is a key benefit of enterprise-level automatic load testing technologies.
However, with automated load testing, the majority of firms will monitor the following metrics:

Response Times
Automatic load testing mainly measures response time. When a user submits a request, how much time does it take for the system to reply? (Users are likely to abandon a site if its response time exceeds 10 seconds.)

Throughput
Data transmission and reception rates are measured by throughput. Common units of measure in load testing include hits per sec (hps) and transaction per second (tps).

Hardware-Specific Metrics
Load testing involves keeping an eye on things like CPU use, accessible RAM, disk I/O, and other comparable hardware-based operations since slow response times might indicate hardware constraints.

Database
The majority of enterprise-level programs rely on many systems to function. However, the likelihood of encountering a bottleneck grows in proportion to the number of databases. Database reads and writes, together with the total number of open connections, are all quantified by load testing tools.
Apache JMeter and the Python-based Locust tool are compared by subjecting both tools to identical load and traffic conditions in order to determine execution time and throughput.

Table 1: There is a comparison of Apache JMeter, Locust, with HULK Analyzer with respect to testing parameters pertaining to the time spent running in seconds.

Testing Attempt	Apache JMeter	Python based Locust	HULK Analyzer
1	1.432	1.093	1.107
2	1.534	1.291	1.472
3	1.938	1.342	1.420
4	2.208	1.938	2.091
5	3.422	2.819	2.989

Conclusion
We should think about both performance and load testing during the design process as they are both non-functional types of testing. When the number of users increases to millions all at once, performance testing becomes crucial. It is not an easy process to evaluate performance and load. This task needs the assistance of skilled individuals and a competent team. On the basis of the testing-tuning-testing cycle is this testing technique. Load and efficiency testing are both rendered impossible in the absence of appropriate tool support; hence, this article details two tools that are fundamental to the process of load testing web applications. We are also concerned with the method of load testing as well as the many characteristics, aims, and activities of performance testing since it is crucial to check whether tools are suited for particular application and performance objectives.

Future Work
There are huge opportunities in the field of web application security that we must investigate and resolve. Scholars, businesses, and government agencies in the domains of software testing as well as forensic auditing can delve into a wide range of topics, such as biometric incorporated entry for secured internet pages, building trust for secured websites, understanding vulnerabilities in web uses with integrity, and identifying suspects in web applications through deep learning. The entire procedure of software testing may be enhanced on numerous aspects with the incorporation of new tools as well as open source libraries. This will allow for the distribution of software goods with improved performance and no bugs.

References
1. Lee S, Ali J, Roh B. Performance Comparison of Software Defined Networking Simulators for Tactical Network: Mininet vs. OPNET. In: 2019 International Conference on Computing, Networking and Communications (ICNC); c2019. p. 197-202.
2. Michael N, Ramannavar N, Shen Y, Patil S, Sung J. CloudPerf: A Performance Test Framework for

Distributed and Dynamic Multi-Tenant Environments. In: Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering; c2017.
3. Picha KJ, Quintana CP, Glueck AC, Hoch MC, Heebner NR, Abt JP. Reliability of 5 Novel Reaction Time and Cognitive Load Protocols. J Sport Rehabil. 2018;27(5):1-4.
4. Jindal A, Podolskiy V, Gerndt M. Performance Modeling for Cloud Microservice Applications. In: Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering; c2019.
5. Miao X, Jin X, Ding J. A hierarchical load balancing parallel computing approach for finite element structural analysis. Chin Sci Bull. 2017;62:1430-1438.
6. Vergara I, Mateo-Abad M, Saucedo-Figueredo MC, Machón M, Montiel-Luque A, Vrotsou K, *et al*. Description of frail older people profiles according to four screening tools applied in primary care settings: a cross sectional analysis. BMC Geriatr. 2019, 19.
7. Wiik MK, Fufa SM, Andresen I, Brattebø H, Gustavsen A. A Norwegian zero emission neighbourhood (ZEN) definition and a ZEN key performance indicator (KPI) tool. IOP Conf Ser Earth Environ Sci. 2019, 352.
8. Vögele C, Hoorn AV, Schulz E, Hasselbring W, Krcmar H. WESSBAS: extraction of probabilistic workload specifications for load testing and performance prediction-a model-driven approach for session-based application systems. Softw Syst Model. 2016;17:443-477.
9. Lenka RK, Rani Dey M, Bhanse P, Barik RK. Performance and Load Testing: Tools and Challenges. In: 2018 International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE); c2018. p. 2257-2261.
10. Zhang Y, Meisner D, Mars J, Tang L. Treadmill: Attributing the Source of Tail Latency through Precise Load Testing and Statistical Inference. In: 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA); c2016. p. 456-468.
11. Noor J, Hossain MG, Alam M, Uddin A, Chellappan S, Al Islam AB. svLoad: An Automated Test-Driven Architecture for Load Testing in Cloud Systems. In: 2018 IEEE Global Communications Conference (GLOBECOM); c2018. p. 1-7.
12. Nandal V, Solanki DK. Performance Testing on Web-based Application using LoadRunner; c2018.
13. Umar A, Abbas M, Rehman S. Metrics and tools that are available for testing scalability. In: 12th IADIS International Conference Information Systems 2019; 2019.
14. Ding B, Xu L, Yang J, Yang R, Yuejin D. The Effect of Stage-Diffuser Interaction on the Aerodynamic Performance and Design of LP Steam Turbine Exhaust Systems. In: Volume 8: Microturbines, Turbochargers, and Small Turbomachines; Steam Turbines; c2018.
15. Péan TQ, Costa-Castelló R, Fuentes E, Salom J. Experimental Testing of Variable Speed Heat Pump Control Strategies for Enhancing Energy Flexibility in Buildings. IEEE Access. 2019;7:37071-37087.