Analytics Performance Load Test

Harsh Verma
SDET Software Tools and Engineering Department
Hike Messenger
35 Upper Mohalla, near Rajesh Electricals Raipur
Dehradun, India(248008)

Arun Krishnan
Director Software Tools and Engineering Department
Hike Messenger
House No 30, Bhagyasree Royale, Rachenahalli Main Road.Tanissandra Bangalore

Sivasubramanian
Specialist SDET Software Tools and Engineering Department
Hike Messenger
Flat No 306, Roopen Comforts,
3rd main A Road, NGR Layout, Madiwala Post Bangalore

ABSTRACT

Any application developed for millions of users requires being tested for its performance, scale, UI and backend aspects. Some of the existing load test tools like JMeter, Tsung, MQTT Malaria do not provide a generic interface to mimic all real-time scenarios. The existing tools can help in achieving a high scale load but do not help in following a user journey of an application at a vast scale where user journey refers to follow actual action path user perform while using the application in real time. The solution to this is the analytic approach used to perform user journey in the application, which helps in identifying bottleneck and stress point easily. In this paper, the focus is on an analytical load testing tool and approach used for load testing various services supporting any protocol. The tool approach comprises robust modular design pattern, scalable for incorporating any new test plan for a load test. A dynamically synchronize report shows graphical stats generated after every load test execution which can be analyzed to detect endpoints. The proposed approach helps to stress the backend/DB/Caches both vertically & horizontally supporting any protocol. The work demonstrates architectural design pattern, generation and execution of test plan and load simulation by analytic Load Runner tool. The framework can mimic the real-time user scenarios and generate specific load.

General Terms

Analytics Performance Load Test, Analytical Stress Test.

Keywords

Analytics Load Test, analytics performance testing, automated testing and tools, performance and load testing, stress testing, testing mindset, and psychology

1. INTRODUCTION

There are various open source tool available for load test [1] [2] like J-meter[3], MQTT-Malaria[4], TSung[5] however, when it comes to processing a user journey at vast scale none of them provide the current market need. This is where the role of analytic performance test comes into play.

The paper demonstrates the use of an analytic performance Load Testing approach as a solution to mock real user scenarios while load testing various services supporting nexus of protocols. It introduces a framework that satiates need of simulating real-time user journey with the desired load as part of CI pipeline.

The motivation for this tool is to incorporate various real-time user scenarios based on Analytic Systems and stress all the back-end systems (DB's, Caches, Queues etc). It's designed and developed with the aim to simulate user journey and to distribute the high load over each server component based on given probability for each action performed by the end-user. Furthermore, to find the threshold point at which the back-end systems starts degrading, the latency of various working services, DB's and Cache's performance.

As mentioned for any application over million users, billions of messages and profuse requests, one needs to know how well the system can cater all the functionalities constantly without any hassles. The current world needs states to mock exact user journey at a vast scale based on analytics. The data points in the analytic systems are dynamic, so you need a system that could easily couple with analytic and make required changes before load generation. In addition to this, to facilitate custom reporting i.e. tracing various parameters (CPU Percentage, Load average), throughput, desired latency the solution HikeRunner simulates the real production user's flow based on the production analytic data.

2. ARCHITECTURAL DESCRIPTION OF ANALYTIC LOAD TOOL

The designed tool is primarily divided into four independent modules:

• **Generator**: This core module continuously generates the test plans(Real Time User flow) based on the given probabilities for each action derived from analytics. Test plan also contains information like data required to execute the particular task. The output of generator will be in the form of JSON format and carried forward to subsequent modules.

• **Executor**: This module executes the tasks given by generator and provides desired load for given test plan by executing the tasks concurrently.

• **User cache pool:** Maintain separate in-memory user pools data for different tasks.

• **Reporter**: Stat-D dynamic report allows finding response rate, breakpoints, and latency.

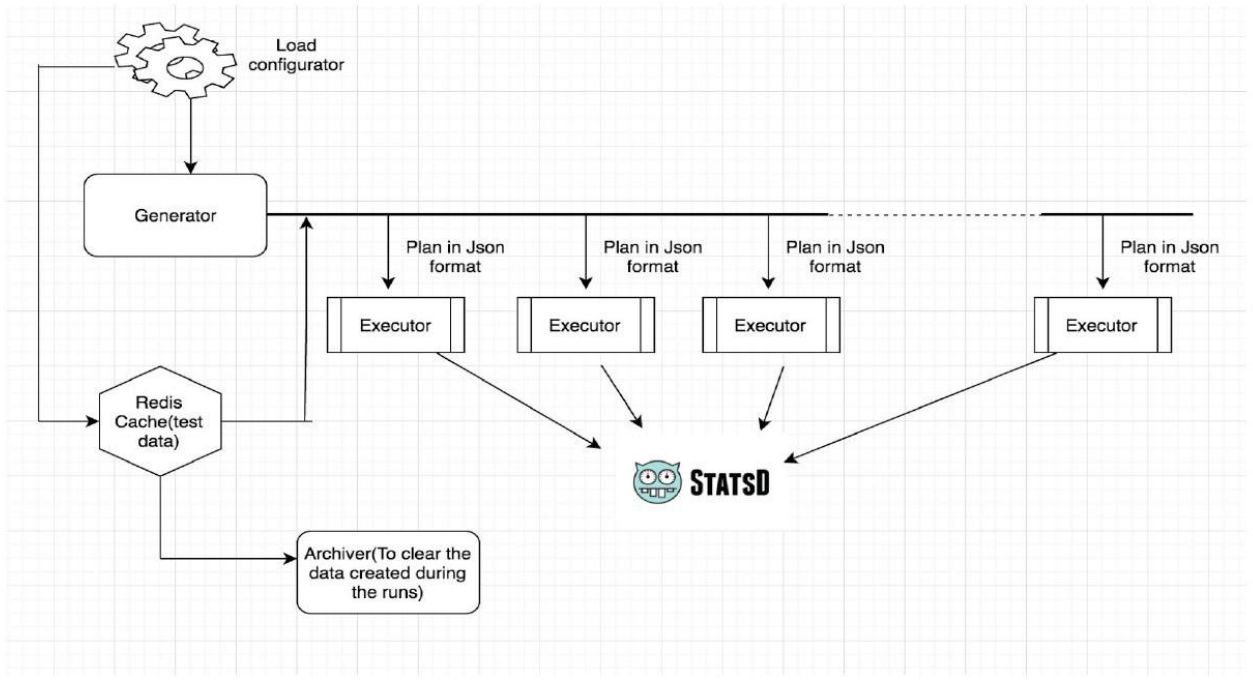


Fig 1: Architectural Diagram of Analytic Load Test Tool

3. FLOW DIAGRAM FOR LOAD GENERATION:

Generator -> User Cache Pool -> Executor -> Reporter

3.1. Sample JSON generated by generator module for user actions:

```
{"offlineMode":false,
"task":"LastSeenTask",
"data":[{"considerUsers":2,"activeUsers":2}],
"actions":[{"action":"LastSeen"}]}
```

Fig 2: Sample JSON Generated by First Module (Generator)

4. KEY FEATURE OF ANALYTIC APPROACH:

- Robust modular design pattern, which helps generate real-time user test plans based on the given run-time dynamic analytic probabilities.
- Simulates various user scenarios to solve load testing for MQTT [6] and HTTP [7] protocol and can be extended to cover other protocols for load/stress test.
- Stats-D graphical report generation for comparing latency, response time and throughput after every load test execution, In addition to this, the same can be used to detect threshold point.
- Independent generator, executor modules help in dynamic and faster execution of generated plans making modules independent of languages to be used (Java, go, Python).

5. CI WITH JENKINS GIT-HOOK AND ANALYTIC LOAD TEST:

Hooking the load test tool with Jenkins facilitated auto execute load tests on latest code-base whenever desire changes are done on the master code.

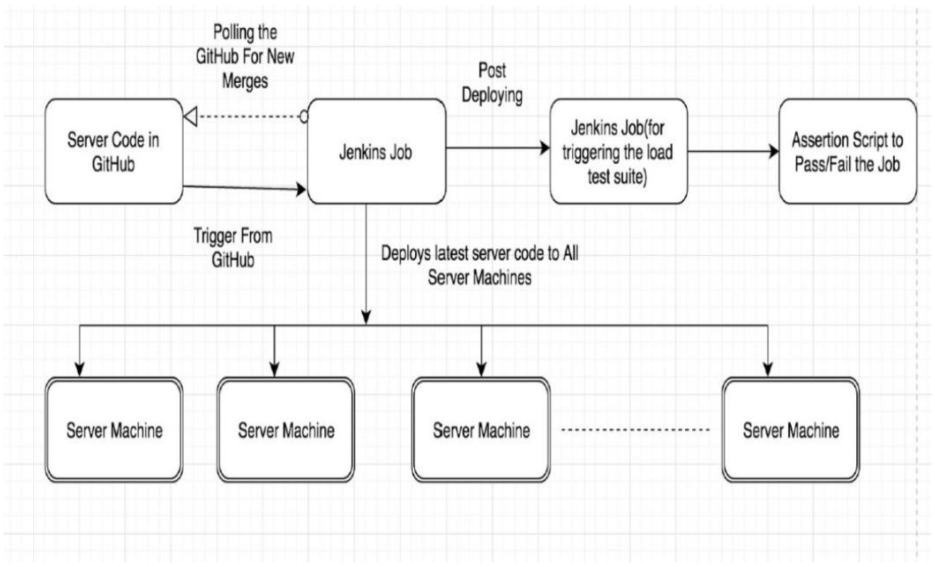

Polling the GitHub For New Merges
Server Code in GitHub
Jenkins Job
Post Deploying
Jenkins Job(for triggering the load test suite)
Assertion Script to Pass/Fail the Job
Trigger From GitHub
Deploys latest server code to All Server Machines
Server Machine
Server Machine
Server Machine
Server Machine

Fig 3: Jenkins CI Web-hook Architecture with Analytics Load Tool

6. RESULTS AND GRAPHICAL REPORT

The monitoring and analysis of the system during load testing helps in gathering useful information about the application; The infrastructure manager can take various decisions and reach a proper conclusion based on these results. Analytic performance test can also help raise alarm for more complex problems that arise in the application at the initial level. This helps in ensuring that more complex problems will not rise. The result during this type of testing can be end-user response time, CPU response time and memory statistics. These results give system data to perform on it for better response on the application ([8] Zhu et al., 2010; Zhao and Shum, 2006; Menascé, 2002; [9] Weyuker and Vokolos,2000). The Stat-D Report generated via CI tool Jenkins after every load test execution helps to analysis throughput, latency, threshold point and benchmark successive release [10]. Some of the samples reports and numbers are displaced in below graphs that specify the number of connections and requests over the analytic user journey over a vast scale.

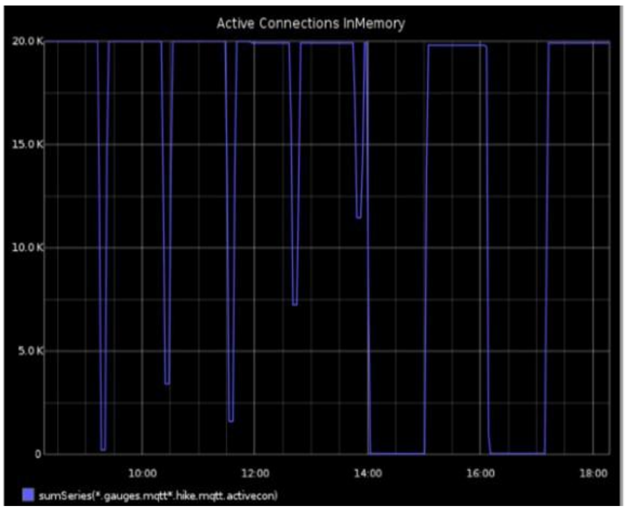


Fig 4: Sample Output of Stats-D Report after load execution displaying traffic count/sec

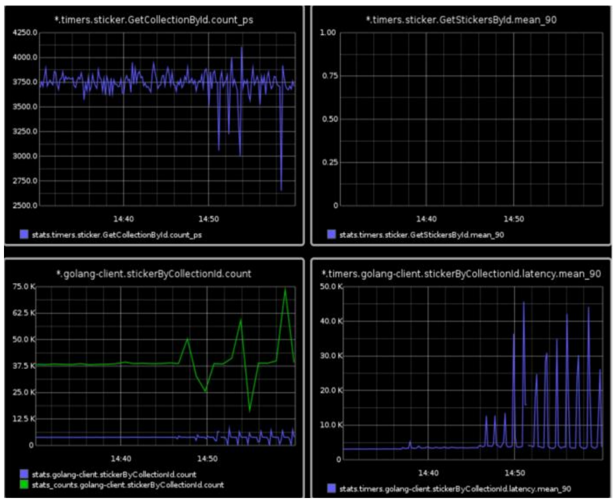


Fig 5: Sample Output of Stats-D Report after load execution displaying latency

The Detail Analysis covers various metrics constituting CPU utilization, latency, network, consumption, memory to trace any bottleneck easily.

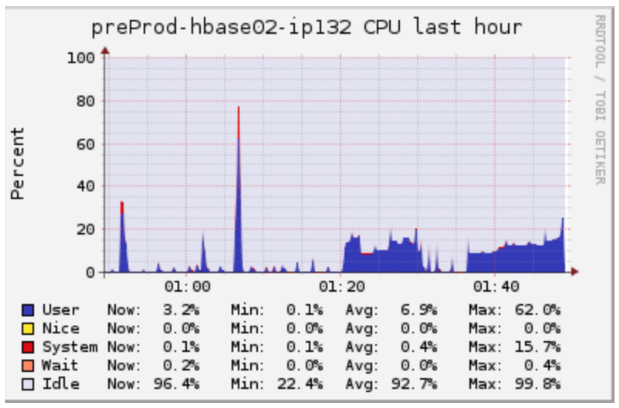
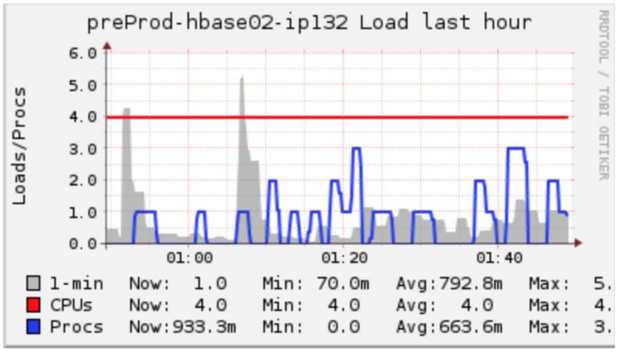


Fig 6: CPU Utilization Result after load test execution

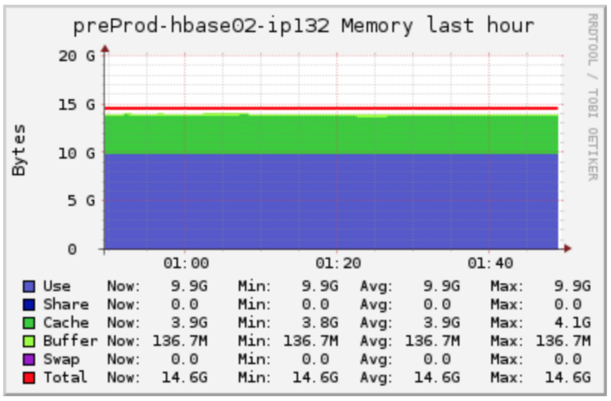
6.1 Some important metrics to be taken care includes :

- Throughput
- CPU Utilization
- Memory Consumption
- Network Bandwidth
- Network traffic In/Out from machines
- Latency
- Load Average on system



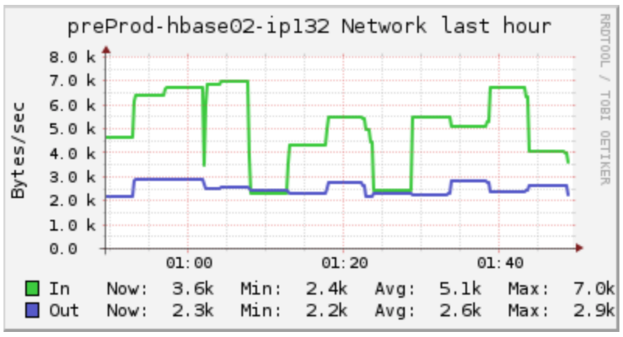
```
■ 1-min   Now:  1.0     Min: 70.0m   Avg:792.8m   Max:  5.
■ CPUs    Now:  4.0     Min:  4.0    Avg:  4.0    Max:  4.
■ Procs   Now:933.3m    Min:  0.0    Avg:663.6m   Max:  3.
```

Fig 7: Load Average stats after load test execution



```
■ Use     Now:   9.9G    Min:   9.9G   Avg:   9.9G   Max:   9.9G
■ Share   Now:   0.0     Min:   0.0    Avg:   0.0    Max:   0.0
■ Cache   Now:   3.9G    Min:   3.8G   Avg:   3.9G   Max:   4.1G
■ Buffer  Now: 136.7M    Min: 136.7M   Avg: 136.7M   Max: 136.7M
■ Swap    Now:   0.0     Min:   0.0    Avg:   0.0    Max:   0.0
■ Total   Now:  14.6G    Min:  14.6G   Avg:  14.6G   Max:  14.6G
```

Fig 8: Memory Consumption stats after load test execution



```
■ In    Now:  3.6k   Min:  2.4k   Avg:  5.1k   Max:  7.0k
■ Out   Now:  2.3k   Min:  2.2k   Avg:  2.6k   Max:  2.9k
```

Fig 9: Network transaction stats after load test execution

7. CONCLUSION

Once the desired user journey via load tool has been tested at vast scale. CPU Utilization, Throughput and hits per second observed during system under test can be found. Mocking user journey based on analytic helps to find easy bottleneck points, performance degradation at various layers of DB/cache/service. Some of the assertion scripts can be used later to find threshold point and generate alerts via CI tools like Jenkins. Moreover, Infrastructure related decisions can be taken into consideration based on generated results. Whether the system meets its performance targets (response times, resource consumption) with the profuse volume of users, transaction volumes and rates can be easily traced. In this way, Analytic performance test helps to validate resource consumption over time, but one needs to take into consideration gradual memory leaks that lead to plummeting of available memory over time. To make the performance of load tool up to date system needs to get sporadic garbage clean up for logging system i.e. log files which tend to grow over time after a certain point and may cause local system issues: insufficient disk space. To sum up, an application that has been developed and tested using analytic performance tool approach is extremely helpful and makes the system ready to streamline user actions in production. The future scope of above analytics approach is immensely useful in the field of data analytics and software testing. It provides a bridge to merge analytics with performance testing. The Approach is helpful to streamline load testing by performing user’s footprint on any application.

8. REFERENCES

[1] Load Multiplier web site, https://loadmultiplier.com/

[2] Jitsi Hammer project page, https://github.com/jitsi/jitsihammer

[3] Apache JMeter™ http://jmeter.apache.org/

[4] Attacking MQTT systems with Mosquittos (scalability and load testing utilities for MQTT environments) https://github.com/remakeelectric/mqtt-malaria

[5] Tsung https://en.wikipedia.org/wiki/Tsung

[6] MQTT Netty: https://github.com/jk-5/netty-mqtt

[7] Http protocol: https://http2.github.io/

[8] Zhu, K., Fu, J., Li, Y., 2010. Research the performance testing and performance improvement strategy in the web application. In: 20102nd International Conference on Education Technology and Computer, vol. 2.

[9] Weyuker, E.J., Vokolos, F.I., 2000. Experience with performance testing of software systems: issues, an approach, and case study.IEEE Trans. Software. Eng. 12, 1147—1156.

[10] Menascé, D., 2002. Load testing of websites. IEEE Internet Computing.6 (4), 70—74.