ISSN:0975-9646

Enhanced Grinder Framework with Scheduling and Improved Agents

Shilpa Sharma[1], Meenakshi Sharma[2]

[1]*M.Tech. C.S.E,*
[2]*H.O.D CSE*
[1,2]*S.S.C.E.T, Pathankot, India*

***Abstract*: - Researchers are still trying to find effective ways to test web application. There are many techniques and tools for web application testing. In this dissertation, we have enhanced grinder performance tool with scheduling and improved Agents. The Grinder makes use of a powerful distributed Java load testing framework that allows simulation of multiple user loads across different "agents" which can be managed by a centralized controller or "console". A grinder agent/worker processes exits once Grinder Console process ends session. We have added scheduler which works as a service on Grinder agent systems. This add-on will help users to start grinder from console itself. Grinder does not provide the way for distributing the agents so they must be deployed and started manually in all machines. There is no way that these agents keep running at back end. In enhanced grinder load increases in steps as done in J meter. Previous Grinder does not allow addition of heterogeneous agents to a grinder test, we will be removing this limitation by making Grinder agents use specified configuration in agents' properties while seeming to grinder Console. Enhanced grinder allows heterogeneous workers capability.**

***Keywords*:** Load testing for web application, Performance testing, Grinder.

1. INTRODUCTION

1.1 Software Testing

Software testing is an important stage in software life cycle. Testing is a process of evaluating a system or its components with the intent to find that whether it satisfied requirements or not. This activity result in the actual expected and difference between their results. In simple words testing is executing a system in order to identify any gaps, errors or missing requirements in contrary to the actual desire or requirements. Different companies have different designation for people who test the software on the basis of their experience and knowledge such as software tester, software quality assurance engineer and QA Analyst etc. Testing is applied to find bugs and used to calculate software bugs density. In typical software projects, the percentages of software testing workload are about 40%.

1.2 Software performance testing

Performance testing is generally executed to determine how a system or sub-system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage. In software engineering, performance testing is in general testing performed to determine how a system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage. Performance testing is a subset of performance engineering, an emerging computer science practice which strives to build performance into the implementation, design and architecture of a system. Performance testing can be performed across the web, and even done in different parts of the country, since it is known that the response times of the internet itself vary regionally. It can also be done in-house, although routers would then need to be configured to introduce the lag what would typically occur on public networks.

1. 3 Load testing for web application

As the web application become popular, it is an urgent issue to how to test them. Web applications are popular due to the ubiquity of web browsers, and the convenience of using a web browser as a client. A web application performance tool (WAPT) is used to test web applications and web related interfaces [2]. It helps to identify the maximum operating Capacity of an application as well as any bottlenecks and determine which element is causing degradation. As the network technology development and increase of web application users pay more and more attention to system performance. Since the web application mixed lots of technology such as HTML, Java, Java Script, database network and EJB result in the testing for web application becomes more complex and difficulty, so the load testing for web application is suggested. Load testing lets you measure your website's QOS performance based on actual customer behavior.

1.4 Grinder

The Grinder is a Java load testing framework that makes it easy to run a distributed test using many load injector machines. Grinder is an free open source desktop application designed to load test functional behavior and measure performance [1]. Grinder scripts are written in Jython programming language. The Grinder is a Java load testing framework making it easy to orchestrate activities of a test scripts in many processes across many machines using graphical console application.

Key features:

1) **Generic Approach** Load tests anything that has a Java API. This includes common cases such as HTTP web

servers, SOAP and REST web services, and application servers (CORBA, RMI, JMS, EJBs), as well as custom protocols.

2) **Flexible Scripting Test** scripts are written in the Jython and Closure language.
3) **Distributed Framework** A graphical console allows multiple load injectors to be monitored and controlled and provides centralized scripts editing and distribution.
4) **Mature HTTP Support** Automatic management of client connections and cookies. SSL Proxy aware. Connection throttling. Sophisticated record and replay of the interaction between a browser and a web site.

1.4.1 The Grinder processes

The Grinder is a JAVA framework for running test scripts across a number of machines. It is free open source under a BSD-style License and supports large scale testing using distributed load injector machines. The main selling point of Grinder however is that it is lightweight and easy to use. With grinder there are no licenses to buy or large environments to set up. The Grinder framework is comprised of three types of *process* (or program): worker processes, agent processes, and the console.

The responsibilities of each of the process types are:

- **Worker processes**
  - Each worker process can run many tests in parallel using a number of worker threads.
- **Agent processes**
  - Long running process that starts and stops worker processes as required.
  - Maintains a local cache of test scripts distributed from the console.
- **The Console**
  - Coordinates the other processes.
  - Collates and displays statistics.

Agent processes

When an agent is started, it attempts to connect to the console. If it can connect, it will wait for a signal from the console before starting worker processes. Otherwise, the agent process will start a number of worker processes as specified by its local grinder. Properties file. If the network connection between the agent and the console is terminated, or the console exits, the agent will exit.

Worker processes

Worker processes are started by a controlling agent process. The agent process passes each worker a set of properties that control its behavior.

2. RELATED WORK

Researchers and practitioners are still trying to effective ways to model and test Web applications. This paper proposes a system-level testing technique that combines test generation based on finite state machines with constraints. We use a hierarchical approach to model potentially large Web Applications. The approach builds hierarchies of Finite State Machines (FSMs) that model subsystems of the Web applications, and then generates test requirements as subsequences of states in the FSM. These subsequences are then combined and refined to form complete executable tests. The constraints are used to select a reduced set of inputs with the goal of reducing the state space explosion otherwise inherent in using FSMs [18]. In order to deliver quality assured software and avoid potential costs caused by unstable software, software testing is essential in software lifecycle. Load testing is one of the testing types with high importance. It is usually accompanied by performance monitoring of the hosting environment. In the case of web applications which are today widely used, one fact is obvious: most of web applications are public and used by vast number of users, which are making a considerable traffic load on hosting environments and web applications [19] . Load testing of IT projects attempts to ensure that the application meets SLA before it is actually launched in the production environment. But, limitations of load testing are its applicability for large number of users, lack of knowledge about the exact production workload characteristics etc. This paper proposes an extrapolation strategy for load testing results which allows one to obtain throughput and response time of an application for large number of users [20]. Load testing and performance monitoring become facilitated with existing tools aimed for load testing and performance monitoring. Web application script crashes and malformed dynamically generated web pages are common errors and they seriously impact the usability of Web applications [21]. Static analysis tools for webpage validation cannot handle the dynamically generated pages that are Ubiquitous on today's Internet. They present a dynamic test generation technique for the domain of dynamic Web application, utilizes both combined concrete an symbolic execution. Load testing of IT applications faces the challenge of providing high quality test results that would represent the performance in production like scenarios, without Incurring high cost of commercial load testing tools [22] . It would help IT projects to be able to test with a small number of users and extrapolate to scenarios with much larger number of users. Such an extrapolation strategy when applied to mixture of application workloads running on a shared server environment must take into consideration application characteristics (CPU/IO intensive, memory bound) as well the server capabilities. WS-TaaS, a load testing platform for web services, which enables load testing process to be as close as possible to the real running scenarios. In this way, we aim at providing testers with more accurate performance testing results than existing tools. WS-TaaS is developed on the basis of our existing Cloud PaaS platform: Service4 All [23] .

First, they briefly introduce the functionalities and main components of Service4All. Second, we provide detailed analysis of the requirements of Web Service load testing and present the conceptual architecture and design of key components. Third, they present the implementation details of WS TaaS on the basis of Service4All. Finally, they perform a set of experiments based on the testing of real web services, and the experiments illustrate that WS- TaaS can

efficiently facilitate the whole process of Web Service load testing. An extrapolation strategy that analyses a system workload mix based on its service demand on various resources and extrapolates its performance using simple empirical modeling techniques. Moreover, its ability to extrapolate throughput of an application mixture even if there is a change in the mixture, can help in capacity planning of the system [24.

Different performance testing tools Neo Load, WAPT and Loadstar are compared in terms of their different performance parameters results in different browsers [25]. Performance parameters results generated by these performance testing tools have been evaluated and analyzed. A comparative study of open source web service testing tools with technical overview and features. Comparison is made on several quality factors including response time, throughput, and usability. Tools are evaluated by collecting the sample web services and collecting the test results [26].

3. METHODOLOGY USED

The Grinder is Java based load testing framework that makes it easy to run a distributed test using many load injector machines. There are a few limitations in grinder that make it hectic to use these tools. We shall update grinder code to update these problems and make grinder easy to use tools and add some of flexibility to its usage.

1. **For Grinder Agents**: A grinder agent/worker processes exits once Grinder Console process ends session. We shall be adding scheduler which works as a service on Grinder agent systems. This add-on will help users to start grinder from console itself.
2. **For Grinder Agent**: Grinder does not provide the way for distributing the agents so they must be deployed and started manually on all machines. So removing this limitation by increase the load in steps.
3. **For adding heterogeneous agents**: Grinder does not allow addition of heterogeneous agents to a grinder test; we will be removing this limitation by making Grinder agent use specified configurations in Agent properties while seeming to Grinder Console.

4. RESULTS AND DISCUSSION

The methodology of Grinder tool is implementing in Eclipse Platform. It is written mostly in java. It can be used to develop application in Java and by means of various plug-in other programming language including Ada, C, C++, COBOL, FORTRAN, Python, Haskell, JavaScript etc.

1. To add test scheduling capability in grinder.

Figure 4.1 is the actual screenshot of the code in which scheduler is added, which works as a service on Grinder agent systems.

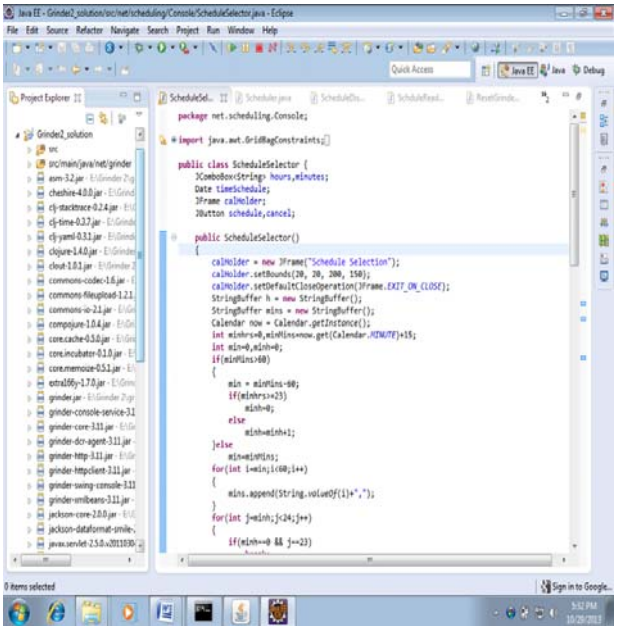

Figure 4.1: Screen shot of code for adding scheduler

In Figure 4.1 we will add scheduler for 15 minute. This add-on will help users to start grinder from console itself.

Output:

Figure 4.2 is screen shot of Agent output when scheduler is added. Each worker process sets up a network connection to the console to report statistics. Each agent process sets up a connection to the console to receive commands, which it passes on to its worker processes. The console listens for both types of connection on a particular address and port. By default, the console listens on port 6372 on all local network interfaces of the machine running the console.

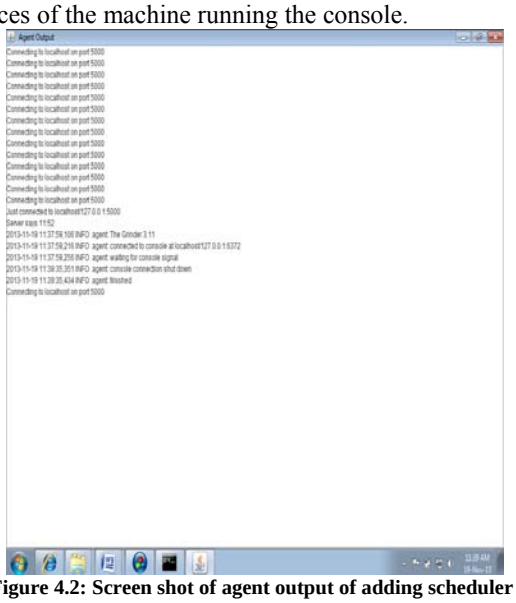

Figure 4.2: Screen shot of agent output of adding scheduler

2. To Enhance Grinder to increase load on application in steps

Figure 4.3 is the actual screen shot of the code in which load is increased on application in steps. In Enhanced Grinder load increased gradually.

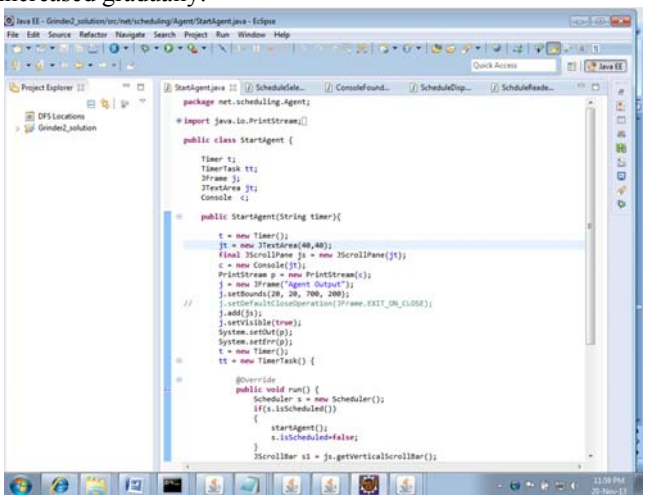

Figure 4.3: Screen shot of code for increasing the load in steps

Output

Figure 4.4 is Screen shot of agents output of increasing load in steps . The differnce between adding users is 20 in which load increased gradually. By default, the console listens on port 6372 on all local network interfaces of the machine running the console.

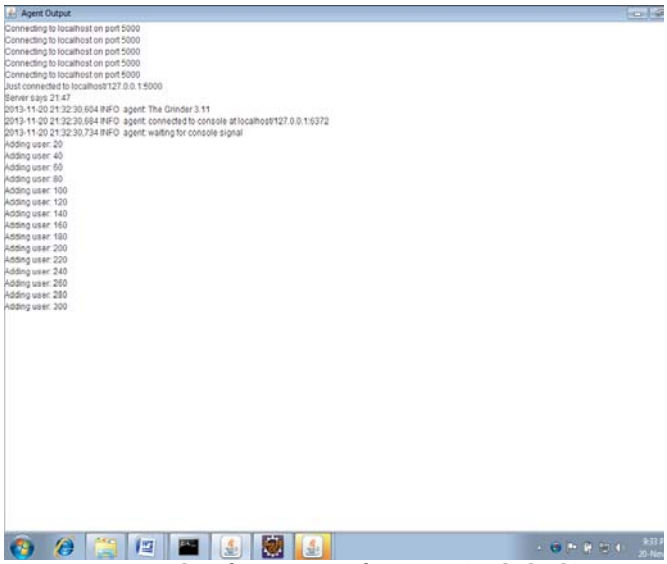

Figure 4.4: Screen shot of Agent output for Increasing the load in steps

3. To add heterogeneous workers capability.

In order to add the hetrogenous workers capablities I changed the updated grinder properties. The Grinder worker and agent processes are controlled by setting properties in the grinder. Properties file. All properties have default values.

Enhanced Grinder can run in different configuration. In which abstract layer is add between agent and cosole so that proper communication is done.

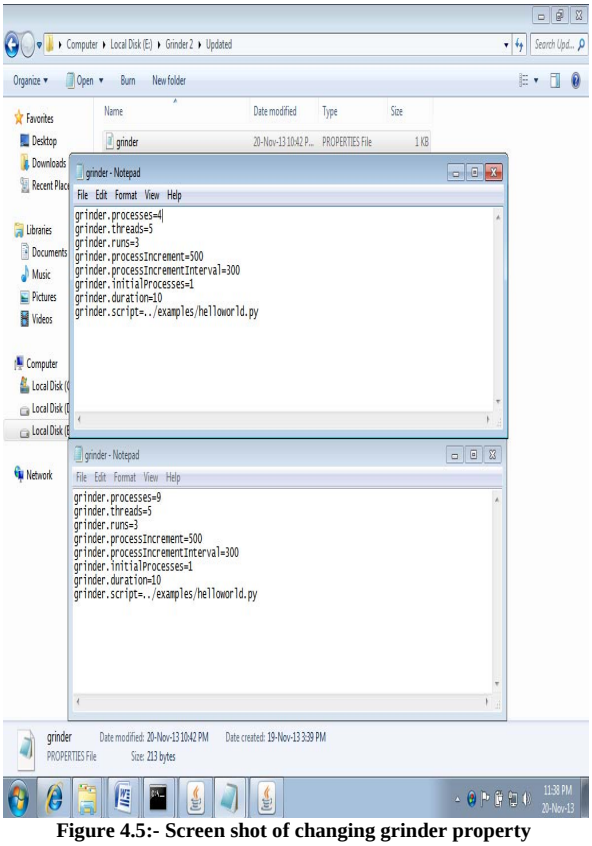

Figure 4.5:- Screen shot of changing grinder property

In Figure 4.5, we change configuration property of Grinder 3 times, every time agent will be started.

Output

Figure 4.6 is the snap shot of heterogeneous workers capability in grinder. Enhanced grinder can run in different configurations. In which abstract layer is set up between agent and console for communication.

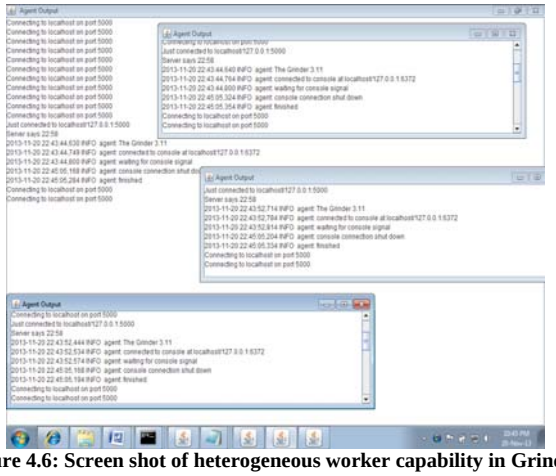

Figure 4.6: Screen shot of heterogeneous worker capability in Grinder

5. CONCLUSION

Enhanced Grinder testing tool helps in easily managing testing suites and requires lesser manual intervention. This help in reducing efforts for performance testing. In Grinder 3, there is no scheduling possibility and load can be increased gradually. When an agent is started, it attempts to connect to the console. If the network connection between the agent and the console is terminated, or the console exits, the agent will exit. Enhanced grinder removes this limitation by adding scheduling capability. The scheduler will start Agents automatically before test starts. Grinder agents need to be started manually each time starting a load testing suite. It does not provide the way for distributing the agents so they must be deployed and started manually on all machines. There is no way that these agents keep running at back end. So removing this limitation by increasing the load in steps as done in Jmeter. Jmeter is also open source Java application designed to load test functional behavior and measure performance. Grinder does not allow heterogeneous workers capability. So remove this limitation by making Grinder agent use specified configurations in Agent properties while seeming to Grinder Console. Enhanced grinder can work in different configuration from which heterogeneous workers can add. Enhanced Grinder allows arbitrary branching and looping and makes test result directly available different test paths to be taken depending on the outcome of each test. It can use the full power of Jython or Closure to create dynamic requests of arbitrary complexity. Enhanced Grinder tool, the most required features is support for performing load test process steps with emphasis on recording, distributing tests, HTTPS and AJAX support.

6. FUTURE SCOPE

Grinder testing tool supports java based applications only, it can be enhanced to increase scope of tool and use with other technologies also. This tool is sufficient for any Web application load testing although higher level of technical expertise is needed to properly use them. Regarding the results analysis, the Grinder provides just log files, so in order to get more though analysis Grinder analyzer tool can be used.

REFERENCES

1. The Grinder website, http://grinder.sourceforge.net/
2. Load Testing. http://en.wikipedia.org/wiki/Load testing.
3. H. Ashram, ―Performance extrapolation in discrete-event systems simulation,‖ Int. Journal of Systems Science, vol. 27, no. 9, 1996, pp. 863-869.
4. Nageswaran , S. Performance Tools Comparison [Online]. Testing reflections .com. Available: http://www.testingretlections.com/, 2005.
5. S. Erlbaum, G. Rothermel, S. Karre, and M. Fisher, "Leveraging user session data to support Web application testing," IEEE Transactions on Software Engineering, vol. 31, no. 3, 2005, pp.187-202.
6. Andrews, J. Offutt, and R . Alexander, "Testing web applications by modeling with FSMs," Software and Systems Modeling, vol. 4, no. 3, pp. 326– 345, Jul. 2005.
7. Barber, S. Choosing Performance Testing [Online]. Testingreflections.com. Available: http://www.testingreflections.coml node /view /3939, 2006.
8. Nageswaran, S. Performance Tools Comparison [Online]. Testing reflections. Com. Available: http://www.testingretlections.com/node/view /1756 , 2006.
9. Barber, S., Beyond Performance Testing, Rational Developer Network. Perf Test Plus Inc. Florida, United States of America, 2007.
10. Deng Xia openg, etc. "Progress in Testing for web Applications [J], Journal of Computer Research and Development, 44(8):1273- 1283, 2007.
11. Microsoft Corporation. Quantifying End-User Response Time Goals - in Performance Testing Guidance for Web Applications. Microsoft Press. United States of America, 2007 .
12. S. Sam path, S. Sprenkle, E. Gibson, L. Pollock, and A. Souter, "Applying concept analysis to user-session-based testing of Web applications," IEEE Transactions on Software Engineering, vol. 33, no.10, 2007, pp.643-658.
13. Vinod, P, "Open Source & Commercial Performance Testing Tools" [Online]. Accenture. Available:
14. Molyneaux, I, Choosing the Right Performance Testing Tools - in the Art of Application Performance Testing. O'Reilly Media. United States of America, 2009.
15. Pu yumig, Xu mingna," Load Testing for Web Application", the 1st International conferences on information Sciences and Engineering (ICISE2009).
16. A. Khanapurkar , S. Malan, and M. Nambiar, ―A Framework for Automated System Performance Testing,‖ in Proceedings of the Computer Measurements Group's Conference, 2010.
17. J. Križaniü, A. Grguriü , M. Mošmondor , P. Lazarevski, "Load testing and performance monitoring tools in use with AJAX based web application", Ericsson Nikola Tesla d.d. Krapinska 45, Zagreb, Croatia, MIPRO 2010, May 24-28, 2010.
18. Dr. S. M. Afroz , N. Elezabeth Rani and N. Indira Priyadarshini, "Web Application– A Study on Comparing Software Testing Tools", International Journal of Computer Science and Telecommunications, Volume 2, Issue 3, June 2011.
19. Subhasri Dutta gupta, Manoj Nambiar," Performance Extrapolation for Load Testing Results of Mixture of application", UK Sim 5th European Symposium on Computer Modeling and Simulation", 2011.
20. S. Dutta gupta, and R. Mansharamani, ―Extrapolation Tool for Load Testing Results‖, Proc. of Int. Symp. on Performance Evaluation of Computer Systems and Telecommunication Systems, SPECTS 2011.
21. M. Yan, H. Sun, X. Wang and X. Liu, Building a TaaS Platform for Web Service Load Testing, IEEE International Conference on Cluster Computing 2012, Sep 2012.
22. Minzhi Yan, Hailing Sun, Xu Wang, Xuedong Liu, "WS- TaaS: A Testing as a Service Platform for Web Service Load Testing", IEEE 18th International Conference on Parallel and Distributing Systems, school of Computer science and Engineering. Beijing University, Beijing, China, 2012.
23. Muhammad Dhiauddin Mohamed Suffiani, Fairul Rizal Fahrurazi, "Performance Testing: Analyzing Differences of Response Time between Performance Testing Tools", in proceeding of International Conference on Computer & Information Science (ICCIS), 2012.
24. Sneha Khoria and Pragati Upadhyay, "Performance evaluation and comparison of software testing tools", VSRD International Journal of Computer Science & Information Technology, Vol. 2 No. 10, October 2012.
25. Rina, Sanjay Tyagi," A Comparative Study of Performance Testing Tools", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 5, May 2013.
26. Shariq Husain, Zhao shun Wang, Ibrahim Kalil Tour and Abdoulaye Diop, "Web Service Testing Tools: A Comparative Study", IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 1, No 3, January 2013.