Model-Based Performance Testing in the Cloud Using the MBPeT Tool

Fredrik Abbors, Tanwir Ahmad, Dragos Truscan, Ivan Porres
Department of Information Technologies, Åbo Akademi University
Joukahaisenkatu 3-5 A, 20520, Turku, Finland
[Fredrik.Abbors, Tanwir.Ahmad, Dragos.Truscan, Ivan.Porres]@abo.fi

ABSTRACT
We present an approach for performance testing of software services. We use Probabilistic Timed Automata to model the workload of the system, by describing how different user types interact with the system. We use these models to generate load in real-time and we measure different performance indicators. An in-house developed tool, MBPeT, is used to support our approach. We exemplify with an auction web service case study and show how performance information about the system under test can be collected.

Categories and Subject Descriptors
D.4.8 [Software]: performance—measurement, modeling, monitors

Keywords
Model-Based Performance Testing, Workload Model, Load Generation, Probabilistic Timed Automata.

1. INTRODUCTION
The goal of performance testing is to validate the system under test (SUT) in terms of responsiveness, stability, and resource utilization when it is put under a certain synthetic workload [3]. The synthetic workload should mimic the real workload as generated when real users interact with the system, as closely as possible [5]. Traditionally, the synthetic workload is generated by simulating virtual user (VU) behavior with scripts or pre-recorded scenarios. However, real users do not behave like static scripts. Furthermore, these scripts or scenarios can be tedious to create and maintain. Therefore, we propose an approach, supported by the MBPeT tool [1], where load is generated from abstract graphical models describing VU behavior. As such, the models are on the one hand easier to create and update, and on the other hand easier to comprehend compared to scripts, due to their abstract and graphical nature, respectively.

2. PROBABILISTIC TIMED AUTOMATA
The MBPeT tool uses models described as probabilistic timed automata (PTA) [4] to generate synthetic workload. A PTA consists of locations and transitions. The transitions can be labeled with three different values: a probability value, an action, and a clock. The probability indicates the chance of that transition being taken. The action describes what action to take when the transition is fired, and the clock indicates how long to wait before firing the transition.

Figure 1 shows an example of a PTA describing a user profile for an online-auctioning system. Transitions have associated either a probability (e.g., 0.1) or a clock and an action to be executed. The former transitions can be used for making probabilistic choices between different paths in the graph. The later are used to send the actions corresponding to a given transition to the SUT. When a response is received, the clock is reset to zero and the next transition is fired. With the help of the probability values defined in the PTA some actions (or sequence of actions) are more likely to be chosen by MBPeT over another actions, whenever a choice is encountered in the PTA. Using PTA allows for a certain level of randomness in the generated workload which makes the later closer to the real workload.

3. MBPET TOOL
The MBPeT tool has a distributed architecture in which a master node controls and distributes the load generation on several slave nodes. Before the slaves start to generate load, the master node loads the PTA model, validates its consistency and initializes a test database with all the necessary test data, e.g., user names, password, form data, files, etc., that is needed to generate meaningful load.

The slave nodes are activated on-the-fly if more generation capacity is needed. If a slave node gets saturated, e.g., having the utilization local resources over a specified threshold, it notifies the master, while maintaining the current load capacity. The master then decides to continue load generation on another slave. The approach is especially beneficial if the slave nodes have heterogenous computation power, eliminat-
The MBPeT tool has been built to run in a private cloud or in a public one, such as the Amazon EC2 cloud. Since the master node does not require much computational power, it can run locally on a laptop, while the slave nodes can be set up to run in the cloud.

4. GOALS OF THE DEMONSTRATION

The goals of the demonstration is to show that using abstract models minimizes the effort of creating the workload profiles. Although the implementation the adapter requires an initial effort it can be afterwards reused without changes. Having available graphical models makes easier to visualize and change the workload profiles. With respect to load generation, using PTAs allows us to model the user profiles and generate synthetic load, via probabilistic-guided load generation, that mimics closer the real one workload. We also want to show how load is generated from the workload models and explain how the tool works. We plan to show the tool running the master node on a laptop and have the slave nodes running in the Amazon EC2 cloud. However, if it is not possible to connect properly to Amazon, we will demonstrate the tool running locally on two machines. During the demonstration we intend to show the tool in action on two different case studies, an auctioning web service and a web-based file storage service. We have done a preliminary evaluation of the MBPeT against the JMeter tool and we could conclude that for identical test configurations and load profiles, the results were similar in terms of throughput (number of actions generated per minute). We plan to briefly discuss our conclusions during the demo as well.

5. REFERENCES

[1] F. Abbors, T. Ahmad, D. Truscan, and I. Porres. MBPeT: A Model-Based Performance Testing Tool. 2012 Fourth International Conference on Advances in System Testing and Validation Lifecycle, 2012.

[2] Amazon. Amazon elastic compute cloud (amazon ec2). http://aws.amazon.com/ec2/. Retrieved: October 2012.

[3] D. Ferrari. On the foundations of artificial workload design. In Proceedings of the 1984 ACM SIGMETRICS conference on Measurement and modeling of computer systems, SIGMETRICS ’84, pages 8–14, New York, NY, USA, 1984. ACM.

[4] M. Jurdziński, M. Kwiatkowska, G. Norman, and A. Trivedi. Concavely-Priced Probabilistic Timed Automata. In M. Bravetti and G. Zavattaro, editors, Proc. 20th International Conference on Concurrency Theory (CONCUR’09), volume 5710 of LNCS, pages 415–430. Springer, 2009.

[5] J. Shaw. Web Application Performance Testing – a Case Study of an On-line Learning Application. BT Technology Journal, 18(2):79–86, Apr. 2000.

[6] Sun. Faban harness and benchmark framework,. Online at http://java.net/projects/faban., 2013.

[7] The Apache Software Foundation. JMeter. Online at http://jmeter.apache.org/. Retrieved: October 2012.