Performance Testing of a Real-World System

Mehra Nouroz Borazjany
Advanced Research Center on Software Testing and Quality Assurance
Department of Computer Science
The University of Texas at Dallas
Richardson, Texas
mehra@utdallas.edu

***Abstract*—One major concern for most companies is to assure that the capacity of their system meets their customers' performance requirements. In this short paper we describe our approach to performance testing of an asset tracking system that provides asset tracking to very large number of customers. For the system under test, we collect data from the available production system and then evaluate the capacity of the system under existence and future expected load. We used jmeter to simulate the communication of tracking devices with the system platform and gather performance metrics such as system response time, resource utilization, and throughput. This helped us to validate whether the system can scale to handle the future expected capacity. This being done by identifying the bottlenecks and allocating enough resources. Additionally, performance testing enabled us to verify the system recovery time after a spike load**

***Keywords*—Performance Testing, Capacity Planning, System Assurance**

I. Introduction

Companies measure the Quality of their Services (QOS) in terms of resource utilization, throughput and response time. Expected QOS assures more satisfied customers which leads to more business opportunities. We need to identify how a system responds to a specified set of conditions and inputs, which is the process of performance testing.

The main goal of performance testing is to identify how well your system performs in relation to your performance objectives. These objectives are often specified in a service level agreement. Additionally, performance testing helps define the conditions under which the system will fail, how it will fail, and what indicators should be monitored to notify us of an impending failure.

There are three major types of performance testing that share similarities yet accomplish different goals. Load testing, verifies system behavior under normal and peak load conditions. Stress testing, leads to evaluate our system's behavior when it is pushed beyond the normal or peak load conditions. Capacity testing helps us identify a scaling strategy to determine whether we should scale up or scale out based on our future growth.

It is important to have a performance testing plan for our system to ensure that it can perform under expected and peak load conditions. Ideally, the result of such testing can determine when and how to upgrade resources and scale the system sufficiently to handle increased capacity. Thus, we allocate adequate resources where they will generate the most benefit.

II. QOS Measurement

Quality of service (QOS) is the overall performance of the system under test. In order to measure the QOS, several quantitative aspects of the system are studied, such as response time, resource utilization, and throughput.

One of the key quality factors of a system is response time. We need to measure end-to-end response time and find the speed of the system under load.

Another key factor is resource utilization. We measure the resource utilization in terms of the amount of CPU, RAM, network I/O, and disk I/O that system consumes under normal or peak loads.

Throughput of a system is also another key factor. Throughputs are rate of services that a system can handle or process per unit of time.

III. TOOLS

There are several tools available to simulate load. We can simulate load in terms of users, connections, data, and etc. Also, they help us gather performance-related metrics such as response time, requests per second, and performance counters from remote server computers. In our project, we used jmeter [1] to simulate the traffic and gather metrics.

In addition, we need monitoring tools to monitor the status of various network services, servers, and other hardware. In this project, all of our servers are monitored by Zabbix[2].

IV. The System Under Test

The system under test provides asset tracking to very large number of customers. The front-end of the system is a web application that our customers can login and track their assets. They can use geofencing, over speed alerts and other exception notifications to ensure their assets are where there should be and safe even when they are not actively managing them.

The back-end is Apache ActiveMQ [3] server, which communicate with a Gateway. Asset tracking devices send information such as IMEI, device type, gps location, service type, and event type to the Gateway. The Gateway transfers the messages to the relevant ActiveMQ server. (Fig. 1)

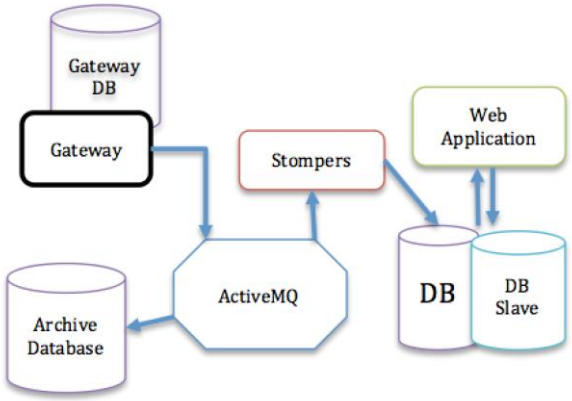


Figure 1: Overall System Architecture

Like any other testing, we need to have a testing plan and strategy based on the nature of the test and the architecture of the system under test. In the plan, we need to specify the goal and scope of the test, our approach, all the possible preparations, all risks, and also some technical details.

The goal of our performance testing for asset tracking system is to validate that the system can scale to handle the expected capacity planned for the coming year. We want to protect our system against crashes and slowdowns on instantaneous peak and verify the system recovery time after the spike load. Thus, we need to answer the two most common questions. First, how many concurrent users are going to use this system. Second, how many devices will communicate with the platform, which basically means how many messages per second needs to be handled.

The first approach is to simulate the activity of devices communicating with the platform. The second approach is to simulate the activities of the end-users logging in and using the web application. The third approach is to simultaneously simulate both devices and the end-users to measure capacity levels at key points within the architecture. In this paper we will only describe the first approach.

Some preparations are also needed. First, because we don't want to run our testing on production system, we need to prepare a staging system, which mirrors the production system. Therefore, tests against it can confirm the capacity of production environment.

Second, we need to characterize the message flow and user activity. We can do this by observing the production archive data. If the system is new, then we can characterize it based on different use case scenarios.

Next, we need to populate the database according to our test plan. For example, to simulate the activities of a certain number of devices, users or accounts, we need to create test accounts/devices within the application infrastructure.

Next, we will prepare the test scripts according to the plan and run them using simulation tools. We then capture the capacity levels using operational monitoring tools.

The risk involved in this project is that some of the network and system resources are shared between our production and our staging systems. As a result, if any of these shared resources prove to be a capacity bottleneck, the production system could be impacted during the testing.

Last but not least is to analyze the results. Depends on our results and intermediate findings, we may require multiple runs. Finally we are ready to report our findings.

V. RESULTS

As it is discussed, according to the testing plan, the goal is to verify whether the system can scale to handle the expected capacity planned for the coming year. The first approach is to simulate the activity of devices communicating with the platform. As it is shown in Fig 2, we sent 74k messages/sec (spike load) to the ActiveMQ server. As a result, system was able to receive all the messages and hold them in the queue without any loss or crash.

Also as it is shown in the system architecture (Fig. 1), stompers are reading the messages from the queue, process them and store them into the database. It is shown in the Fig. 2, that stompers are able to process 70 messages per second. Moreover, it shows the recovery rate of the system. For example, it took the system 1000s to process all the messages and recover.

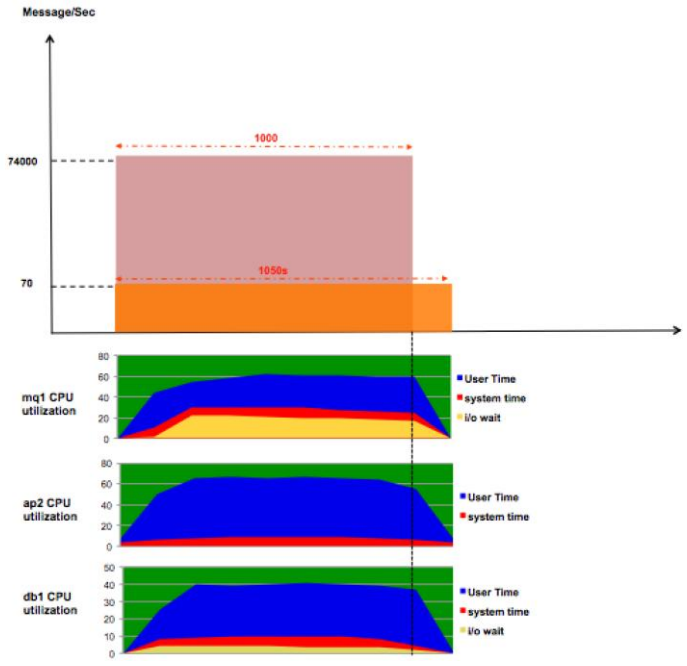


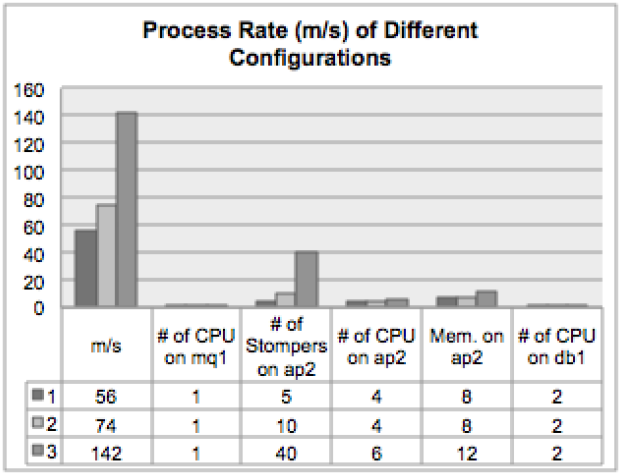
Figure 2: Capacity testing

The actual capacity of the system was 50 m/s. In order to increase the capacity of the system to 150 m/sec, we need to make some changes to the system configuration. For example,

we could increase the number of stompers from 5 to 10, which would increase the process rate to 70 m/s.

Fig. 3 displays some of the modifications with their process rates. As it is shown, in order to increase the capacity to 150 m/s, we need to increase the number of stompers to 40, and allocate more CPUs and memories to the server.

The second approach is to simulate the activity of the end-users logging in and using the web application. We believe that there are more studies describing the web application performance testing such as Kunhua Zhu et al.[9] and Yanyan Lu et al.[10]; therefore, in this short paper we only explained the back-end performance testing.



	m/s	# of CPU on mq1	# of Stompers on ap2	# of CPU on ap2	Mem. on ap2	# of CPU on db1
1	56	1	5	4	8	2
2	74	1	10	4	8	2
3	142	1	40	6	12	2

Figure 3: Configurations

REFERENCES

[1] http://jmeter.apache.org/

[2] http://www.zabbix.com/about.php

[3] http://activemq.apache.org/

[4] D.A. Menascé and V.A.F. Almeida, "Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning", Prentice Hall, Upper Saddle River, N.J., 2000.

[5] P.J. Denning and J.P. Buzen, "The Operational Analysis of Queuing Network Models," ACM Computing Surveys, vol. 10, no. 3, Sept. 1978, pp. 225-261.

[6] D.A. Menascé and V.A.F. Almeida, "Capacity Planning for Web Services: Metrics, Models, and Methods", Prentice Hall, Upper Saddle River, N.J., 2002.

[7] D.A. Menascé et al., "A Methodology for Workload Characterization of E-commerce Sites," Proc. First ACM Conf. Elec- tronic Commerce, ACM Press, New York, 1999, pp. 119-128.

[8] S. Vinoski, "Web Services Interaction Models: Part I: Current Practice," IEEE Internet Computing, May/June 2002, pp. 89-91.

[9] Kunhua Zhu, Junhui Fu, and Yancui Li, "Research the performance testing and performance improvement strategy in web application", 2nd international Conference on Education Technology and Computer (ICETC), 2010.

[10] Yanyan Lu, Haiyan Wu, Yingxue Wang, "Web Application Performance Analysis Based on Comprehensive Load Testing", ICWMMN2006 Proceeding

AUTHOR BIOGRAPHY

Dr. Mehra Nouroz Borazjany is a Senior Lecturer in the Department of Computer Science at the university of Texas at Dallas. Additionally Dr. Borazjany is a member of the Software Testing and Quality Assurance Center. She received her Ph.D. degree in May 2013 from the Computer Science and Engineering department at the University of Texas at Arlington. She had the opportunity to be supervised by Prof. Jeff Lei in the Software Engineering Research Center (SERC) Group.

The topic of her PhD research was in the area of automated software analysis, testing and verification. Before joining UTD, she worked as a Performance Test Engineer.