Building a TaaS Platform for Web Service Load Testing

Minzhi Yan, Hailong Sun, Xu Wang, Xudong Liu
School of Computer Science and Engineering
Beihang University, Beijing, China
{yanmz, sunhl, wangxu, liuxd}@act.buaa.edu.cn

***Abstract*—Web services are widely known as the building blocks of typical service oriented applications. The performance of such an application system is mainly dependent on that of component web services. Thus the effective load testing of web services is of great importance to understand and improve the performance of a service oriented system. However, existing Web Service load testing tools ignore the real characteristics of the practical running environment of a web service, which leads to inaccurate test results. In this work, we present 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: Service4All. First, we provide detailed analysis of the requirements of Web Service load testing and present the conceptual architecture and design of key components. Then we present the implementation details of WS-TaaS on the basis of Service4All. Finally, we 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.**

***Keywords*-web services; cloud computing; load testing; testing as a service**

I. Introduction

Most Internet applications usually face constantly changing user requests. Under-estimating user demand may lead to disastrous system failure. In this aspect, load testing is considered to be an effective method to identify the maximum operating capacity of an application as well as any bottlenecks and determine which factor is causing degradation.

In recent years, service-oriented computing has been regarded as a novel way to develop applications and web services are usually adopted as the building blocks. The performance of such an application system is mainly dependent on that of component web services. Thus the effective load testing of web services is of great importance to understand and improve the performance of a service oriented system. Here we identify three challenges that web service load testing faces as follows: (1) Simulation of real characteristics of massive user requests, including real concurrency, diversity of geographical location and system configuration; (2) Elastic provisioning of testing environments, consisting of underlying resources and runtime middleware; (3) Automating the load testing process.

Existing web service load testing tools, such as SoapUI[1] and Apache JMeter[2], support testers to create and execute load testing tasks on a single node or in a LAN environment. However, in practice users coming from different places may access to a web service concurrently. The single-node or small LAN based load testing cannot simulate the real characteristics of massive user requests due to the limitation of node resources. Indeed, as the increasing of concurrent threads in a test node, more and more threads will be put into the waiting queue, thus these threads actually are not executing the test task simultaneously. Hence, it is costly for traditional test methods to test the target web service from geographically distributed nodes to simulate the practical user requests. In addition, traditional test methods require testers to install test tools, build and configure test environment manually, which will increase the cost of load testing and impact the efficiency as well. In a word, existing tools do not meet the challenges of Web Service load testing.

With the rapid development of cloud computing, resource capacity and many system functions are provided as services, like SaaS (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a Service) and so on. Among them, TaaS (Testing as a Service)[3][4] is proposed to provide software testers with cost-effective testing services. One of the services provided by TaaS is the hosting of hardware and software tools for software testing, which means TaaS is usually built on the top of IaaS and PaaS. There are several advantages that can be obtained by incorporating cloud computing into software testing. First, the testing environment, including middleware and underlying infrastructure, can be provisioned automatically in accordance with tester requirements, and testers do not have to concern about where or how the test environment is built. Second, cloud testing supports great elasticity, which is inherited from cloud computing technology[5], and has the ability to provide test environment for large-scale concurrent testing. Third, it is also an outstanding advantage that cloud testing serves testers with the best convenience of simulating geographically distributed requests. Furthermore, cloud testing providers always supply efficient test task management and scheduling mechanism, as well as test results analysis. Last but not least, cloud testing solution can observably reduce the test cost as pay-as-you-go payment model is adopted.

However, to the best of our knowledge, there is no work providing an online service for Web Service load testing yet. This is probably due to the complexity of hosting web service middlewares and testing tools. In this work, we propose WS-

TaaS, a TaaS for Web Service load testing. WS-TaaS is built on the basis of our PaaS platform Service4All, a cloud platform for service oriented software development[6][7]. With WS-TaaS, testers can deploy, execute and monitor test tasks and obtain test results just through simple configurations. Test environment provisioning is transparent to testers and the environment can simulate the real runtime scenario of a web service. It provides an efficient and accurate test service for WS load testing and can substantially reduce the test cost, complexity and time.

II. Related Work

A. Web Service Load Testing

The Apache JMeter[2] supports load testing functional behavior and measuring performance of web services on single node or a LAN. SoapUI[1] is also a Web Service testing tool which supplies the capability of WS load testing, it allows testers to easily create and execute automated load tests. These two test tools both need to be downloaded and installed in a tester's computer, and the test processes can only be executed on one node or a few nodes, so the load amount is limited. However, our work provides WS load testing as a service, and makes the test process more convenient and accurate.

B. Cloud Testing

In academia, Lian Yu[8] proposed a TaaS model and implemented a VM-based cloud testing environment. In this work, the task scheduling, monitoring and resource management problems were discussed. Zohar Ganon[9] presented a method which leverages commercial cloud computing services for conducting large-scale tests of Network Management Systems. All of these works do not pay attention to Web Service load testing in cloud. In industrial area, Load Impact[10] offers load testing and reporting as an online service to e-commerce & B2B sites. SOASTA's CloudTest[11] supports functional and performance cloud testing services for today's web and mobile application. All the products above cannot support Web Service load testing except the cloud-based test tool TestMaker[12]. However, TestMaker is still a desktop software and not provided as a testing service. It needs installation and the test configuration process is troublesome and difficult for testers. The test cloud needs to be configured manually if a tester wanted to run a test task in Amazon EC2. That is, he has to define the number of required test nodes and lots of other information, which fails to reflect the important feature of elasticity in cloud. However, in our testing environment, testers only need to configure a few information while all the other works can be finished automatically.

III. Design of WS-TaaS

A. Requirements of WS-TaaS

To design a cloud-based Web Service load testing environment, we need to analyze its requirements and take advantage of the strong points of cloud testing. Taking above aspects into account, we conclude that the following features should be guaranteed while building WS-TaaS:

1) Transparency: The transparency in WS-TaaS is divided into two aspects: (a) Testers have no need to know exactly where the test nodes are deployed. (b) When the hardware environment is ready, the testing middleware should be prepared automatically without tester involvement.

2) Elasticity: All the test capabilities should scale up and down automatically commensurate with the test demand. In WS-TaaS, the required resources of every test task should be estimated in advance to provision more or withdraw the extra ones.

3) Geographical Distribution: To simulate the realistic runtime scenario of a web service, WS-TaaS is required to provide geographically distributed test nodes to simulating multiple users from different locations.

4) Massive Concurrency and Sufficient Bandwidth: As in WS load testing process the load span can be very wide, so WS-TaaS have to serve massive concurrent load testing. Meanwhile, the bandwidth need to be sufficient accordingly.

B. Conceptual Architecture

In a cloud-based load testing environment for Web Service, we argue that these four components are needed: Test Task Receiver & Monitor(TTRM), Test Task Manager(TTM), Middleware Manager and TestRunner. Figure 1 shows the conceptual architecture of a cloud-based WS load testing system, including the following four main components:

1) Test Task Receiver & Monitor: TTRM is in charge of supplying testers with friendly guide to input test configuration information and submiting test tasks. The testing process can also be monitored here.

2) Test Task Manager: TTM manages the queue of test tasks and dispatchs them to test nodes in the light of testers' intention, then gathers and trims the test results.

3) Middleware Manager: It is needed to manage all the TestRunners and provide available TestRunners for TTM with elasticity.

4) TestRunner: TestRunners are deployed on each test node and play the role of web service invoker, and they can also analyse the validity of web service invocation results.

C. Workflow

The workflow of WS-TaaS is divided into five steps, as shown in Figure 2. Firstly, a tester inputs the test configuration of a test task in TTRM and submit it to TTM. Secondly, TTM selects test nodes from available nodes according to

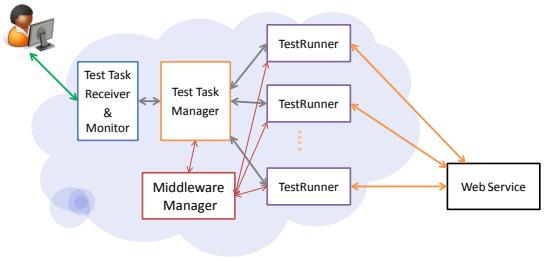

Fig. 1. Conceptual Architecture of a WS Cloud Testing System

their performance, and asks Middleware Manager to provision TestRunners on them. If one TestRunner has been idle for a fixed time span and presently there is no test task designated to it, then Middleware Manager decides to withdraw it and shut it immediately. Thirdly, TTM decides the numbers of concurrent request of each node and divides the test task into subtasks with a specific dispatching strategy, and then sends them to the chosen test nodes. Then, after dispatching the test task, TTM notifies all the participanting TestRunners to start invoking the target service simultaneously. During test task execution, TTRM sends query requests to TTM at a fixed period to obtain the execution state. And TTM also periodically queries the test results from all the involved TestRunners. On receiving a query response, the intermediate test results will be displayed to users. The query process will be continued until the test task finishes.

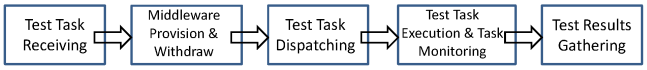

Fig. 2. Workflow of WS-TaaS

IV. Implementation

A. System Architecture

WS-TaaS is a typical three-layer structure of cloud which consists of SaaS, PaaS and IaaS layers. Based on Service4All, we develop three new modules and extend two existing ones in WS-TaaS. Three modules in Figure 3 are newly designed ones, including Web Service LoadTester in SaaS layer, Test Task Manager (TTM) and a new type of Software Appliance(Runtime Middleware in Service4All) TestEngine in PaaS layer. SA(Software Appliance) Manager and Local Agent deployed in every node are also extended to support the registration and management of TestEngine.

B. Key Modules

1) Web Service LoadTester: LoadTester is the implementation of Test Task Receiver & Monitor mentioned in Section III which is responsible for receiving load test tasks from testers and displaying the test results for them.

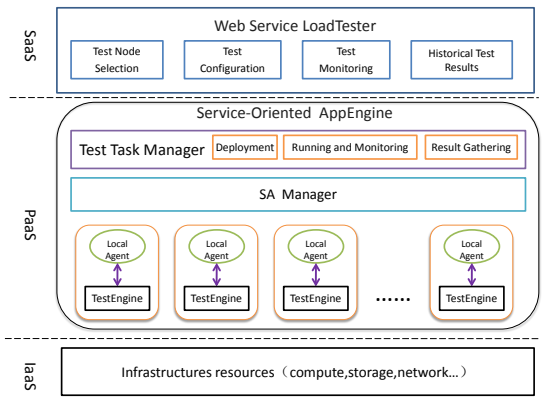

Fig. 3. System Architecture of WS-TaaS

LoadTester provides testers with the possibility of selecting diversely located test nodes. Testers can freely select the needed nodes on the map in the test node selecting view. In test configuration view, after inputting the target web service WSDL address or the service ID in Service4All platform, testers are guided to input the test parameters, then the test message for invoking the target service is generated automatically. LoadTester also receives test configuration information like test strategy and concurrent request number etc. When a test task is submitted to WS-TaaS from LoadTester, it periodically queries the running task, organizes the intermediate data as charts for the monitoring requirement of testers. When a load test task is finished, tester can also save its test results and review them in LoadTester whenever they need.

2) Test Task Manager: Test Task Manager is the core module in WS-TaaS, it is in charge of test task dispatching, test results gathering and preliminary trim.

Test Task Manager receives test task from LoadTester, and then starts the test process according to the test mode described in the test task. For example, static test just needs to be deployed once, while step test needs different times of deployment in the light of test configuration. Before deploying test tasks to test nodes, TTM queries from SA Manager for available TestEngine list. Once the concurrent number is set in a deployment, the corresponding concurrent request numbers of each TestEngine is determined on the basis of our scheduling strategy(The details of our scheduling algorithm are not presented here due to the paper space limitation). After sending test deployment requests to all the participanting TestEngines and receiving their deployment responses, TTM sends the command to start testing, and periodically queries and stores test execution state from these TestEngines. Upon receiving query request for test task state from LoadTester, TTM replies with the latest stored state. It also executes analysis on the intermediate data, e.g., in a maximal test, it needs to analyse the data to determine whether the last deployed test number is the load limit of the target service.

3) TestEngine: TestEngine is the specific SA in WS-TaaS, and it is used to invoke web services.

TestEngine receives test request which contains test message for invoking target service, concurrent number etc. then prepares corresponding number of test threads and reply Test Task Manager with the 'ready' response. When ordered to start the test process, TestEngine activates all the ready threads to invoke the target service simultaneously, it also compares the invoke result with the expected result to ascertain if the test executed in a thread is successful or not, then organizes and stores these information for the test state query from TTM.

4) SA Manager: Middleware Manager is implemented as SA Manager. The functions of SA Manager are extended to manages all types of Software Appliance in Service4All, including web services container, BPMNEngine and TestEngine etc. It is in charge of the registration, querying of SA and decides the deployment and undeployment of all types of SAs according to diverse provision and withdrawal strategy.

5) Local Agent: Local Agent assists SA Manager with managing SAs in a computing node. We extend it to support the operation of TestEngine deployment, undeployment, registration etc. Additionally, Local Agents distributed on each test node detect and collect the performance information of these nodes, including network traffic, CPU and memory usage etc. for comparison of node performance.

V. Experiments

In this section, we use two experiments, in both of which 30 services are chosen as the target services, to compare the performance of traditional single node web service load testing approach(JMeter) with that of WS-TaaS.

(1) *Average Response Time Comparison*: Figure 4 displays the results of three services(S1,S2 and S3) out of the 30 which show that along with the increase of the concurrent request number, the ART(Average Response Time) of JMeter grows quickly while that of WS-TaaS grows at a very slow rate for S1, S2 and S3. Under the load of 250 concurrent requests, the ART of JMeter for S2 is about 6 times the size of the WS-TaaS ART for the same service. As the hardware and network environment are exactly the same, we could make the conclusion that the ART difference under heavy load mainly results from the limited bandwidth and computing capability of the sole node, which leads to the queuing of multiple threads. Nevertheless, instead of queuing, test threads in WS-TaaS can be scheduled more efficiently. And the ART behavior for the other services are the same.

(2) *Error Ratio Comparison*: Figure 5 shows the error ratio comparison of JMeter and WS-TaaS when testing an email address validation web service: ValidateEmail. In WS-TaaS, we define the load limit of a web service as the load amount under which more than 10 percents of the invoking requests are failed. If we also use this determining principle in JMeter, the load limit of this web service will be 210 which is confirmed by further test. However, it can be seen that the load limit given by WS-TaaS is more than 600, and it is about 3 times the size of the former. The great difference between these two results tells that the limited capacity of a single node can distinctly affect the accuracy of test result, and with WS-TaaS, a more accurate result can be obtained. The error ratio comparison results for the other services have similar performance.

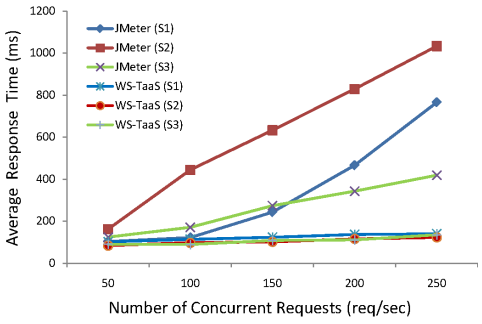

Fig. 4. ART Comparison Under Diverse Load

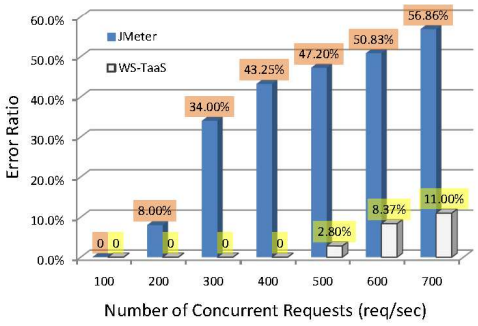

Fig. 5. Error Ratio Comparison

VI. Conclusion

Traditional testing approaches cannot satisfy the load testing requirements of web services, and cloud testing technology can provide a novel and efficient solution for this problem. However, the related work on TaaS platform for WS load testing can barely be seen. Therefore, in the present paper, we identify the requirements of Web Service load testing and propose a testing platform WS-TaaS. The design and implementation details of WS-TaaS based on Service4All are presented. Additionally, two experiments with 30 real web services are performed to illustrate the efficiency and accuracy of load testing using WS-TaaS.

Acknowledgment

This work was supported by China 863 program (No. 2012AA011203), National Natural Science Foundation of China (No. 61103031), the State Key Lab for Software Development Environment (No. SKLSDE-2012ZX-12), the Fundamental Research Funds for the Central Universities (No. YWF-12-LXGY-023), Specialized Research Fund for the Doctoral Program of Higher Education (No. 20111102120016), and CPSF 2011M500218.

References

[1] SoapUI, http://www.soapui.org/
[2] Apache JMeter, http://jmeter.apache.org/
[3] HP Testing as a Service. http://www8.hp.com/us/en/software/software-product.html?compURI=tcm:245-936967
[4] Testing as a Service. http://www.tieto.com/what-we-offer/it-services/testing-as-a-service
[5] L. M. Riungu, O. Taipale, K. Smolander, *Research Issues for Software Testing in the Cloud*, 2^{nd} IEEE International Conference on Cloud Computing Technology and Science, Nov 2010, pp. 557-564.
[6] H. Sun, X. Wang, C. Zhou, Z. Huang and X. Liu, *Early Experience of Building a Cloud Platform for Service Oriented Software Development*, IEEE International Conference on Cluster Computing 2010, Sep 2010.
[7] Service4All, http://www.ow2.org/view/ActivitiesDashboard/Service4All
[8] L. Yu, W. Tsai, X. Chen, L. Liu, Y. Zhao, L. Tang and W. Zhao, *Testing as a Service over Cloud*, 5^{th} IEEE International Symposium on Service Oriented System Engineering, Jun 2010, pp. 181-188.
[9] Z. Ganon, I. E. Zilbershtein, *Cloud-based Performance Testing of Network Management Systems*, IEEE International Workshop on Computer-Aided Modeling Analysis and Design of Communication Links and Networks, Jun 2009. pp. 1-6.
[10] Load Impact, http://loadimpact.com
[11] SOASTA, http://www.soasta.com/
[12] TestMaker, http://www.pushtotest.com/cloudtesting