Cloud-based Performance Testing of Network Management Systems

Zohar Ganon and Itai E. Zilbershtein, *Member, IEEE*

***Abstract*—Network Management Systems are often challenged by the need to manage networks comprised of a very large number of elements. Effective testing of performance, capacity and stability of these management systems often requires significant and, at times, cost-prohibitive investment in equipment and computing resources. This paper presents a method which leverages commercial cloud computing services for conducting large-scale tests of Network Management Systems. This method involves instantiation of large numbers of virtual network elements via the cloud computing infrastructure. After the network of virtual elements is up, it can be used for performance testing of the Network Management System. Considerations for evaluating, planning and implementing this method are discussed. Finally, a case study demonstrating the application and effectiveness of this method is presented.**

***Index Terms*—Cloud computing, Communication system operations and management, Computer network management, Large-scale systems, Software testing**

I. Introduction

In recent years, there have been a proliferation of large-scale distributed communication systems. The advent of Voice over Internet Protocol (VoIP), low cost computing elements, and ubiquitous IP networking provided the ability to deploy large scale networks of communication elements. Furthermore, communication protocols such as the Session Initiation Protocol (SIP) [1] enabled the creation of communication architectures in which much of the functionality and complexity of the communication system is embedded in its endpoints. As managed elements grow in functionality, the management complexity - for example, in terms of required configuration parameters, monitoring capabilities, testing and operational facilities for each function - grows accordingly. To effectively manage a large scale network consisting of thousands or tens of thousands of managed elements, a Network Management System (NMS) must be designed to accommodate the scale, distribution and complexity of the managed elements.

Software testing is part of any commercial software endeavor [2]. Developing an effective NMS solution for a large set of managed elements distributed across a network is no different and requires the ability to effectively test that solution.

A test should be able to provide feedback to designers on how well the NMS handles its tasks, with regard to the following aspects:

1. Performance – key application metrics, such as: CPU utilization, network throughput and response time define application performance. For example, how much time performing management tasks would take, considering the expected number of managed elements, as well as the topology, capacity and latency characteristics of the communication network?
2. Capacity – are data structures, storage facilities, and so on correctly sized?
3. Usability – is the user-interface designed to provide effective user interaction for large amounts of elements?
4. Reliability – Does the NMS perform in a reliable manner?

A typical testing strategy for an NMS application is to test it with the actual elements it needs to manage (Fig. 1a). This gives a high degree of assurance that it will function correctly in real-life conditions with these types of elements. Large-scale networks of managed elements present a challenge in that regard: procuring and setting up a test involving a large number of real elements is a costly and lengthy undertaking. Large-scale deployment tests are therefore often attempted using a simulated environment.

Creation of a simulated environment using general purpose commercial simulators (Fig. 1b) is a common industry practice. However, setting up such tools to simulate a large scale network of relatively complex elements presents certain challenges. In particular, using one physical simulator deployed on a commercial server with a single or a small number of CPUs is restricted by the simulator's limited processing capability and memory. At some point adding more simulated elements will impact the simulator's performance. Using general purpose network simulators such as those provided by Gambit Communications [3] and AdventNet [4] introduces additional challenges, such as handling the full gamut of management protocols by which elements might be managed, and simulation of element-specific business logic.

In recent years, advances in operating system virtualization technologies coupled with commercial reasons to consolidate computing services have introduced new ways to provide computing resources. New ways which realize a long-held idea of computing as a utility [5]. The term *cloud computing* is often used to describe "A pool of abstracted, highly scalable, and managed compute infrastructure capable of hosting

Manuscript received February 11, 2009. Revised manuscript received April 15, 2009. This work was supported by Avaya Inc.

Z. Ganon is with Avaya, Atidim Technology Park building 3, Tel-Aviv, Israel 61581 (phone/fax: +972-3-6457658; e-mail: zganon@avaya.com).

I. E. Zilbershtein is with Avaya, Atidim Technology Park building 3, Tel-Aviv, Israel 61581 (phone/fax: +972-3-6457667; e-mail: izilbers@avaya.com).

978-1-4244-3533-3/09/$25.00 (c)2009 IEEE

end-customer applications, that is billed by consumption" [6]. In particular cloud computing services (such as [7]-[12]) can be utilized to create a computing or networking platform of considerable scale and power, in a way that is cost-effective to applications with varying scalability requirements. In some cases, cloud computing is used to provide software services ("Software as a Service" or *SaaS* model, for example [13]). In other cases, development platforms are offered ("Platform as a Service" or *PaaS*, for example [8][9][14]). In yet another type of service, resources such as storage and CPU are provided to the service's users as infrastructure for their own applications ("Infrastructure as a Service" or *IaaS* which is also known as "Utility Computing" [15], for example [7][10][11][12]). These types of services can be combined. For example, IaaS can be used to provide SaaS. One of the advantages of this approach is that undifferentiated tasks (for example, hardware costs, maintenance and utilization) no longer take the majority of time, energy and money [16]. Exact definitions of the different types of cloud computing services are still being debated [17]. In this paper, we will use the term cloud computing to refer to the latter type, namely "Infrastructure as a Service".

This paper presents a method for NMS performance testing which is based on off-the-shelf "Infrastructure as a Service" cloud computing service. We describe our experience using this method for developing and testing a commercial NMS that manages a distributed system consisting of thousands of VoIP private branch exchange (PBX) networked through SIP.

Several companies (for example, BrowserMob, SOASTA and LoadStorm) provide cloud-based performance testing products. These are used to test web applications and are utilized by recording, editing and running either user interactions or HTTP(S) traffic. NMS testing which is discussed in this paper is different in a few aspects. The method suggested therein requires only a minimal infrastructure apart from the one provided by the cloud computing service itself. Furthermore, using emulation agents instead of recorded HTTP(S) traffic has advantages like: writing application level test cases instead of low level scripts, emulation of element-specific business logic and flexibility in the communication protocols that can be used. Additional cloud-based performance testing approaches have been suggested [18], all of which clearly differ and complement the method described therein.

The remainder of this paper is organized as follows: section 2 describes the cloud-based testing method in detail, delves into considerations for its usage and elaborates on its planning and implementation phases. A case study is presented in section 3 that describes the application of section 2 to solve a real world NMS performance testing challenge. The case study includes selected results, efforts summary and gained insights. Concluding remarks then follow in section 4.

II. CLOUD-BASED NMS PERFORMANCE TESTING

A. Method Description

The following method is proposed (Fig. 1c):

1. An element agent, mimicking the behavior of a real managed element with regard to management interaction is developed. Unlike traditional emulation/simulation frameworks, the agent design can be simplified to mimic a single element on a single operating system/computer instance. In some cases the agent can be the original element's software adjusted to run without the underlying hardware.
2. The element agent is stored on the cloud in a reusable format for execution on a virtual server instance.
3. Test scripts instantiate as many copies of the element agent as needed over the cloud computing infrastructure, running each instance on a unique virtual server identified by a unique IP address.
4. The NMS is tested to work against the cloud network.
5. The testing scripts shutdown the virtual servers.
6. Test results are analyzed and any required fixes to the NMS Software are implemented.
7. Steps 3 to 7 can be repeated with gradually increasing number of element agents until the NMS meets its scalability and quality goals.

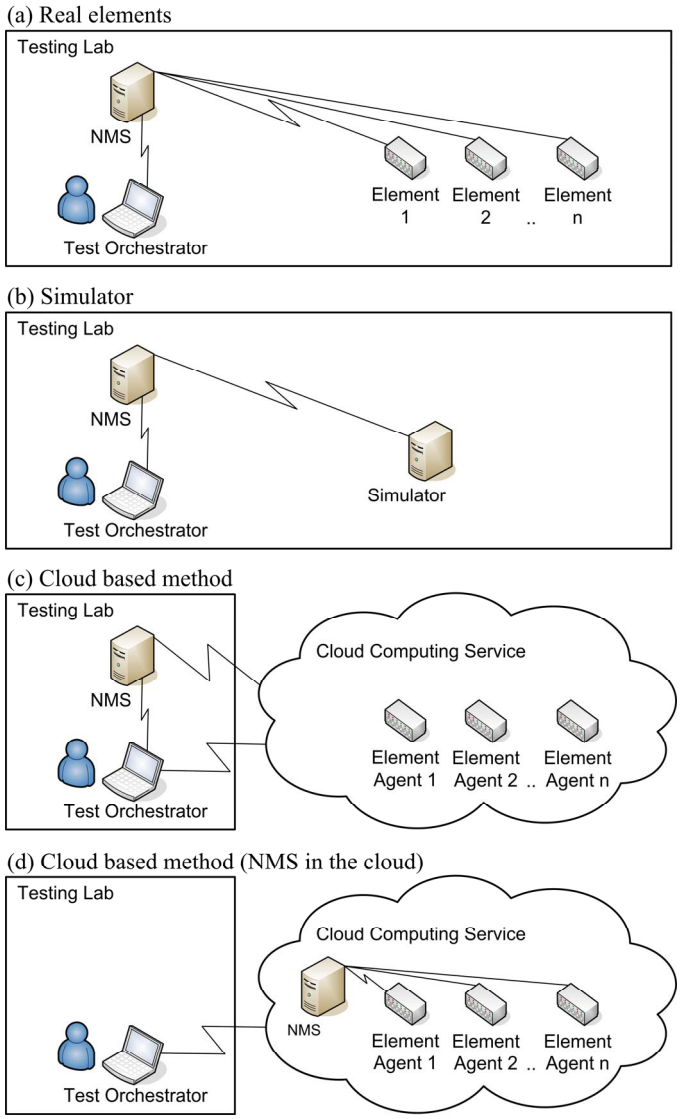

Fig. 1. NMS testing setups

B. Analysis

The cloud-based method has several advantages and disadvantages compared to other approaches (that is, testing vs. real elements or running a simulator).

In terms of validating the correct operation of the NMS at the functional level, testing vs. real elements seems to be a more prudent approach compared to simulation-based approaches, including the cloud-based method. The complexity of the managed elements is such that one can never be sure that a simulator or emulation agent will perfectly mimic the behavior a real element would exhibit. Hence at least some tests should be done against the actual elements the NMS is expected to manage.

Simulation-based approaches have some advantages over testing with real devices:

1. Availability: the simulator might be available before the real element is produced, allowing tests of the NMS to be conducted before the real-element has been developed to the point where it could be tested against.
2. Simulators provide options to test abnormal conditions that might be difficult and costly to test using a real element.

Cloud-based testing is a variation of the simulation-based approach. Compared to procurement/development of a simulator/emulator and execution on servers located in the testing lab, it has the following advantages:

1. Very large scale tests can be easily conducted: At the time of writing of this paper, there are cloud computing services, for example, [7], which provide user accounts with the ability to run thousands of instances. Once an element agent capable of being instantiated on the cloud service had been produced, instantiating thousands of instances is as easy as instantiating a single instance.
2. As the computing resource is rented per-use, a test plan can be devised to make efficient use of the project's testing budget, testing configurations of different sizes and paying just for the actual time and resources used in each test run (as opposed to procurement of a server farm, or renting servers).
3. Elimination of the costs associated with procurement, maintenance and operation of a set of physical servers, including space, power and cooling requirements.
4. An application deployed on the cloud could be easily opened for access by the public. This flexibility can be handy for running limited-time demos of large networks (for example, for partners and potential customers in trade-shows).

Compared to other approaches, Cloud-based testing has certain disadvantages, or issues that need consideration before the method is used to solve a particular testing challenge.

Like other simulation-based approaches, it does not provide the level of certainty that the NMS works correctly as testing with real elements. The level of certainty could grow in cases where the approach of re-using the real element's software as a basis for an emulation agent is applicable. This approach might offer additional advantages to using a wholly synthetic simulator, as a variety of management protocols might be tested and the behavior exhibited by the real element's internal business logic has a higher likelihood of being modeled correctly, without going through lengthy analysis and simulator design phases.

A basic premise of cloud-based testing is that some software elements - the element agents and, potentially, the NMS - will run on infrastructure that is not owned by the testing organization. This has the following implications:

1. Intellectual Property issues: Cloud computing vendors typically provide various security measures to assure that their customers' information (including software files) will not be disclosed to other users of the cloud computing service. And yet, the potential for such a disclosure is probably higher than for tests conducted in a lab that is disconnected from the outside world.
2. Cloud computing providers might have their servers deployed globally. In some cases, there might be export restrictions on certain pieces of software that might place restrictions on where software could be run.

Other points to consider when evaluating cloud-based computing relate to the difference between the computing resources offered by the cloud and that available in other methods that offer greater control for the tester. For example, some operating systems, like VxWorks or Microsoft Windows Embedded for Point of Service, might be problematic to run on off-the-shelf cloud computing infrastructure.

The underlying hardware of a cloud element agent instance would often be different than the real element's hardware. This can have an impact on the response time of the element (an important factor and one which might have a non-linear /complex effect on the NMS' performance). Typically, a network element will have less memory and a less powerful CPU than the most powerful instance available from a cloud computing provider. This is an advantageous situation in comparison to a situation where the real element is faster than the cloud element. The developer who creates the element agent can 'cripple' the element agent (e.g., by introducing delays) in order to have a response time profile which is as close as possible to a real element's response time profile. It is noted that there are NMS performance tests which can be done regardless of the response-time difference between a real element and an element agent. For instance: testing the NMS for the correct amount of thread-pool parallelism, testing the scalability of the NMS' user interface with a lot of real time data and testing the NMS in a more stressful scenario caused by the faster response time.

To conclude – an analysis should be made to evaluate the applicability of the cloud-based testing method for a particular scenario. There are significant benefits for using this method. However, it cannot be used to completely replace traditional tests that use real managed elements.

C. Planning

The planning phase of the cloud-based testing method includes several steps:

1. Determining the focus of the test
2. Planning the test configuration
3. Selecting a Provider
4. Creating a detailed test plan

To determine the focus of the test, one needs to identify which aspects of network management are to be tested, such as

network discovery, configuration, monitoring, and so on. A test might focus on validating a particular design choice, for example, parallelism and thread pool size optimization, scalability of the user interface, or on operational characteristics, for example, the time it takes to perform a certain task.

The NMS might be expected to function in a noisy and unreliable network communication environment, and the test might be required to validate that. To do so, one might plan the deployment of a special network library on the cloud instances for emulating problems such as network delays or packet losses.

Next, one needs to determine what elements would be deployed on the cloud - the element library to be created. The type of elements to be deployed, their software versions and initial configuration is determined at this stage. For instance, one can choose to emulate a typical customer configuration or a configuration that emulates initial installation.

The type and number of elements might influence deployment strategies. One needs to evaluate the possibility of running multiple element agent instances per cloud instance. This option can save cost and enable testing of a number of elements that is much larger than the maximum number of cloud instances that a cloud provider might allow.

The hardware configuration for the NMS should also be decided upon. The tests can help devise those requirements if those are not fixed yet. If those are fixed then the performance tests can help devise usage guidelines for running the NMS in a large scale network setup (for example, performing a management task such as distribution of configuration to a large network should be done in chunks of no more than a specific number of elements at a time).

To test performance independent of the bandwidth and latency characteristics of the connection between the testing lab and the cloud, one can run the NMS itself on the cloud (Fig. 1d). This deployment option might be required if the network elements need to communicate with the NMS in ways that might violate the testing lab organization's firewall access policies.

Selecting a cloud computing provider involves several considerations. A basic requirement is that the provider's infrastructure can support running the element agents and possibly the NMS. These software pieces determine operating system, memory, CPU and storage required on cloud instances.

The provider should support the maximum number of instances the test calls for at an acceptable price point to the project. Legal related matters (for example, compliance with export restrictions, to be determined by the type of software to be tested, by specific government restrictions and by the physical location of the cloud computing infrastructure) need to be complied with.

Providers differ in the type of access they give to instances (for example, SCP/SSH/full vs. limited file system access), as well as in cloud management tools and APIs, and sharing facilities between users of the cloud. These, in turn, might make it easier or more difficult to deploy and operate a test on a particular provider's infrastructure.

Finally, one should write a detailed test-plan that takes into account the choices made through the planning process. We suggest adopting the following gradual multi step approach:

1. Verifying correct operation vs. a small set of real elements.
2. Synthetic population of the NMS databases through testing scripts with data emulating a large deployment.
3. Creating a functional local 'hardware-less' element (for example, running the element's software on a regular Linux machine without all the underlying networking hardware).
4. Uploading the image to the cloud and running one secure and functional cloud element agent instance.
5. Starting with small scale simple tests and gradually increasing the complexity and number of instances involved in the tests.

III. CASE STUDY

In this section we present a case study in which we applied the cloud-based testing method to meet a NMS performance testing challenge. We describe considerations for choosing the cloud-based method, elaborate on the planning and implementation phases and present selected results and amount of invested effort and costs.

Avaya Communication Manager Branch Edition is a Voice over IP telephony switching system typically deployed in small branch offices of larger organizations. An organization might have thousands of these switching elements deployed, and networked together through SIP. The Network Management System, Avaya Communication Manager Branch Edition Central Manager plays a key role in facilitating the deployment and operation of those thousands of telephony systems. It has facilities that allow a network administrator to easily change behaviors (such as a telephone set feature-button mappings) in a consistent manner across those thousands of telephony switches. Our challenge was to develop and test that this NMS can meet its performance requirement – being able to perform management tasks for networks consisting of hundreds and thousands of telephony switching elements.

A. Options

We have been required to validate the design of the NMS vs. a large scale network of hundreds of elements, within a fairly short schedule. We had several options to choose from.

While we have used up to 20 real elements in early stages of the design validation, we have found the option of using hundreds of real elements to test large configurations to be invalid, due to the cost of the hardware, ongoing maintenance and the physical facilities required to support such a deployment in our lab.

Developing a home grown simulator was ruled out because of the estimated amount of effort involved and the limited scalability available when running on a single commercial server. Furthermore, the protocol used for managing the elements was based on Web Services. The general purpose network simulators that we had supported SNMP and were therefore not suitable. Hence, we chose to develop and use the cloud-based method, seeing that it had the potential of meeting all our requirements in a scalable and cost effective way.

B. Planning

We planned to test only the remote configuration aspects of the managed element via its Web Services APIs and not the managed element's telephony services. Among the areas that we planned to test were: NMS' CPU and memory utilization for different tasks including checks for memory leaks, functionality tests, optimal size of the thread pool and successful completion of many configuration jobs running in parallel.

For the element, we decided to re-package the real-element's embedded management software layer as a stand-alone element agent running on a Linux OS with stubs emulating the missing real-elements' software and hardware layers.

In terms of deployment configuration, we planned to have the NMS deployed in our lab and communicate through our company's firewall to the elements deployed on the cloud.

At the time the test was planned, during the first months of 2008, Amazon Web Services (AWS) [7] cloud computing service provided all the required functionality: Linux-based images with full root access, ability to run hundreds of instances, no initial cost and pay for use pricing model. In addition, AWS included extensive documentation, a management console (the Elasticfox Firefox add-on) and a programmatic Web Services-based management API. AWS stood out as one of the leading cloud computing suppliers in the market and for all those reasons it was a natural choice.

We planned to start with one type of element with a very basic configuration and expand the test gradually, testing only the remote configuration aspects and without introducing any noise to the network.

C. Implementation

Prior to running tests on the cloud, we have gone through the two initial steps described in section 2C: verifying correct operation vs. a small set of real elements and synthetic population of the NMS databases through testing scripts. A few problems, such as inefficient database queries or queries running inside loops, were fixed during the second step.

Following these steps, we have verified that a 'hardware-less' element can run and respond as a real element from a configuration perspective. We then created one secure and functional cloud element agent, putting in place some security measures. To setup an emulated Avaya Communication Manager Branch Edition element on the cloud we uploaded the element files to a running instance, tweaked very few files to adjust them to the image's specific Linux flavor and stored the image on the cloud. For instance and cloud administration we have used command line management as well as the Elasticfox Firefox add-on, which greatly simplified the management of AWS. As a safety measure against failure of these management facilities, we have written a simple Java application to enable quick bulk shutdown of cloud instances through a Web Services API.

D. Test Results

We gradually increased the number of tested elements from one element to two elements and then to 20 elements with almost no problems during the process. Then a 300-instance test was performed. This test allowed us to detect a few problems (for example, in our thread-pool design) previously undetected. Through insight gained observing CPU and memory utilization, we have found it beneficial and safe (in terms of reliability) to significantly increase the size of our thread pool. We have found optimizations to be required in some configuration areas, and that the user-interface had a few usability issues related to the scale and performance characteristics of the network.

The goal which was to test the operation of the NMS in a large-scale setup and define specific areas for improvement was achieved. None of the problems mentioned above surfaced during the small-scale testing. The large-scale tests facilitated by cloud computing enabled us to greatly improve the reliability of our NMS software.

E. Effort and Cost

It took a total of two staff-weeks to initiate, develop and conclude the cloud-based testing. This includes mainly the development effort of the element agent but also the study of the cloud provider environment, writing of the management scripts and running the actual tests. The cost of using the provider's cloud for initial tests was less than $10. The 300-instance test took 4 hours to complete. Multiplied by a cost of $0.1 per instance CPU-hour resulted with a cost of $120 for this test.

IV. Conclusions

In this paper, we proposed a method for performance testing of network management systems using commercial cloud computing infrastructure. The method involves preparing and storing images of managed elements on the cloud. Images that can be run later in large numbers using the cloud computing service in order to simulate large scale networks for NMS testing purposes. A case study presenting this method's application to a commercial NMS performance testing challenge testifies to its applicability and cost effectiveness.

The method could also be used on a private cloud created anyhow by the enterprise, a trend predicted in [19]. This would alleviate the legal and security concerns described in section 2.

The appeal of the suggested method may increase as the number of providers offering cloud computing services increases. Price reductions and introduction of additional features (e.g., support of additional operating systems) would also serve to increase its applicability to various scenarios.

Finally, the method could be adapted to test other types of software applications that could benefit from the abstraction, scale and flexibility offered by cloud computing facilities. For instance, while our test focused on the network-management aspect of a communication network, tests focusing on the actual operation of such a network could be conducted as well.

Acknowledgment

We thank our Avaya colleagues, Nir Rachmel and Oren Sanderovich for their technical help during the case study, Dan Romascanu and Gregory Kohll for their comments and Eyal Serero, Yechiel Corn and Amir Gonen for their comments and

continuous support.

REFERENCES

[1] J. Rosenberg *et al.*, "SIP: Session Initiation Protocol." IETF RFC 3261, June 2002.

[2] B. Meyer, "Seven principles of software testing." Computer, vol. 41, no. 8, pp. 99-101, Aug. 2008, doi:10.1109/MC.2008.306.

[3] Mimic Simulator Suite. Gambit Communications, Inc. 2009. http://www.gambitcomm.com/site/products/

[4] AdventNet Simulation Toolkit 7, AdventNet, Inc. 2009. http://www.adventnet.com/products/simulator/index.html

[5] D. F. Parkhill, *The Challenge of the Computer Utility*. US: Addison-Wesley Educational Publishers Inc., 1966.

[6] Forrester Research Glossary, http://www.forrester.com/ER/Glossary

[7] Amazon Web Services, http://aws.amazon.com/

[8] Google App Engine, http://code.google.com/appengine/

[9] Microsoft Azure, http://www.microsoft.com/azure/whatisazure.mspx

[10] GoGrid, http://gogrid.com/

[11] Rackspace, http://www.mosso.com/

[12] FlexiScale, http://www.flexiscale.com/

[13] Google Docs, http://docs.google.com/

[14] Salesforce.com, http://www.force.com/

[15] T. O'Reilly. (2008, October 26). "Web 2.0 and cloud computing." http://radar.oreilly.com/2008/10/web-20-and-cloud-computing.html

[16] W. Vogels, "A Head in the Cloud - The Power of Infrastructure as a Service." O'Reilly MySQL Conference & Expo, Santa Clara, CA, April 15, 2008.

[17] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing." EECS Department, University of California, Berkeley, UCB/EECS-2009-28, February 10, 2009.

[18] J. Varia. (2009, February 24). "Virtual Stress-free Testing in the Cloud." Amazon Web Services Blog. http://aws.typepad.com/aws/2009/02/virtual-stressfree-testing-in-the-cloud.html

[19] J. Brodkin. (2008, November 13). "Private cloud networks predicted for corporate IT." Techworld. http://www.techworld.com/opsys/news/index.cfm?newsid=107032