2011 Eighth IEEE International Conference on e-Business Engineering

Scalability and Performance of Web Applications in a Compute Cloud

Trieu C. Chieu, Ajay Mohindra, and Alexei A. Karve
IBM T. J. Watson Research Center
19 Skyline Drive
Hawthorne, NY, USA
{tchieu, ajaym, karve}@us.ibm.com

***Abstract*—Scalability and performance are key factors to the success of many enterprises involved in doing business on the web. Maintaining sufficient web resources just to meet performance during peak demands can be costly. Compute Cloud provides a powerful environment to allow dynamic scaling of web applications without the needs for user intervention. In this paper, we present a case study on the scalability and performance of web applications in a Cloud. We describe a novel dynamic scaling architecture with a front-end load-balancer for routing user requests to web applications deployed on virtual machine instances with the goal of maximizing resource utilization in instances while minimizing total number of instances. A scaling algorithm for automated provisioning of virtual resources based on threshold number of active user sessions will be introduced. The on-demand capability of the Cloud to rapidly provision and dynamically allocate resources to users will be discussed. Our work has demonstrated the compelling benefits of a Cloud which is capable of sustaining performance upon sudden load surges, delivering satisfactory IT resources on-demands to users, and maintaining high resource utilization, thus reducing infrastructure and management costs.**

Keywords- compute Cloud; scalability; performance; virtualization

I. INTRODUCTION

Cloud Computing [1, 2, 3] has become a popular enterprise model in which computing resources are made available on-demand to users as needed. The vast processing power of a compute Cloud is made possible though distributed, large-scale computing clusters, often in concert with server virtualization software, like VMware ESX Server [4], Xen [5] or KVM [6], and parallel processing. The computing resources may be maintained within the client enterprise, or made available by a service provider. Computing at the scale of a Cloud allows users to access supercomputer-level computing power. Users can consume the enormous and elastic resources whenever they need them. For this reason, Cloud Computing is also described as on-demand computing.

The on-demand model was developed to overcome the common challenge to an enterprise of being able to meet fluctuating demands efficiently. The model of Cloud Computing has evolved by following concepts of utility computing, autonomic computing, grid computing, and software as a service (SaaS) [7]. Utility computing, for example, is an on-demand approach that combines outsourced computing resources and infrastructure management with a usage-based payment structure. Because an enterprise on-demand computing resources can vary drastically from one time to another, maintaining sufficient resources to meet peak requirements can be costly. Conversely, if the enterprise cuts costs by maintaining only minimal computing resources, there may not be sufficient resources to meet peak requirements.

On the other hand, the scalability or the ability to expand and add resources dynamically is critical to the success of many enterprises currently involved in doing business on the Web and in providing information that may suddenly become heavily demanded. While there are many strategies that IT organizations can undertake to serve more customer demands, the way they are designed and implemented can make or break these businesses. Cloud Computing offers a powerful environment to scale web applications without difficulty. In fact, Cloud Computing can provide different resources such as servers, memories, storage and networking on-demand to suit many of the scaling points for web applications. The on-demand nature of Cloud Computing combined with the pay-as-you-go model guarantees that as the application demand grows, so can the resources required to service that demand. In this situation, the capacity equals exactly the demand as long as the application is designed properly and its architecture is amenable to scaling well.

In this paper, we will present a case study on the scalability and performance of web applications in a compute Cloud. To illustrate the powerful scaling capabilities of the Cloud environment, we will introduce a novel scaling scenario for web applications deployed in virtual machines that are created and destroyed in an on-demand fashion. We will first describe a typical Cloud Computing environment. We then present our studies on identifying the scaling indicators for web applications. We will discuss the scalability capabilities of the Cloud and its use of virtualization technologies. We will also introduce our novel architecture design on a scaling scenario with a dynamic scaling algorithm based on number of active users to the web applications.

The outline of the rest of the paper is organized as follows. Section 2 describes the cloud computing architecture and virtualization. Section 3 presents our case study on scalability and performance of a scalable web application. Section 4 illustrates the architecture design of a dynamic scaling scenario for our web application. A scaling algorithm based on number of active users will be described.

978-0-7695-4518-9/11 $26.00 © 2011 IEEE
DOI 10.1109/ICEBE.2011.63

Related work will be discussed in Section 5. Finally, Section 6 concludes the paper.

II. CLOUD COMPUTING, VIRTUALIZATION AND SCALABILITY OF SERVICES

Cloud Computing provides the ability to add capacity as needed, typically with very small lead times. Clearly, Cloud Computing provides a new compelling mechanism for dealing with application services that need to be scalable. A brief introduction of Cloud Computing, its key delivery technology of virtualization, the scalability and the scaling indicators of web applications in a Cloud will be given in the following sections.

A. Cloud Computing

Cloud Computing is a way to deliver services over the network. New advances in virtualization technology [8, 9], processors, disk storage, broadband Internet access and fast, inexpensive and powerful servers have all combined to make Cloud Computing a realistic and compelling paradigm. Cloud Computing basically use virtualization technique to turn computer resources into virtual guest machines. These virtual machines usually reside on some networked physical servers in a hosting environment. However, the virtual guest machines can be moved around, thus breaking direct hardware dependency associated with physical machines. With hardware dependency no longer an issue, the guest system can be insulated from hardware breakdowns, and be automatically moved to another piece of available hardware. A typical Cloud Computing environment with automated provisioning capabilities, advanced virtualization technologies, and virtual machines hosting on physical servers offering Cloud application services to users directly is illustrated in Fig. 1.

In Cloud Computing, new applications are made available by highly efficient virtualized computing resources that can be rapidly scaled up and down in a secure way to deliver a high quality of service. Users can gain access to their applications from anywhere through their connected devices. The user sees only the service and not the implementation or infrastructure required for its delivery. Examples of Cloud services include technology services such as storage, data protection, applications, business processes and even business and consumer services such as email and office applications. Cloud Computing allows users and companies to use the services and storage that they need, when they need them and, as wireless broadband connection options grow, where they need them. Customers can be billed based upon server utilization, processing power used or bandwidth consumed.

As a result, Cloud Computing has the potential to overturn the software industry entirely, as applications are purchased, licensed and run over the network instead of a user desktop. This shift will put data centers and their administrators at the center of the distributed network, as processing power, bandwidth and storage are all managed remotely. The following are typical types of Cloud Computing services depending on nature of offerings:

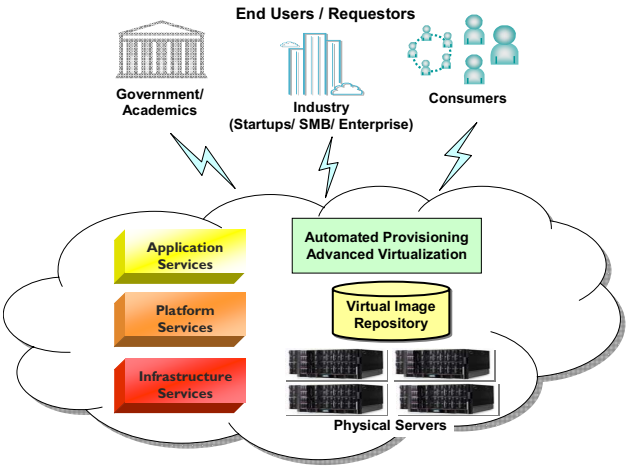

Figure 1. A Typical Compute Cloud Environment.

- ***Application Services*** - Any web application is a Cloud application service in the sense that it resides in the Cloud. Google, Amazon, Facebook, Twitter, Flickr, and virtually every other Web 2.0 application is a Cloud application in this sense.
- ***Platform Services*** - One step up from pure utility computing are Cloud platform services like Google Apps and Google AppsEngine, and Salesforce's force.com, which hide virtual machine instances behind higher-level APIs.
- ***Infrastructure Services*** - Amazon Elastic Compute Cloud (EC2) is a typical Cloud infrastructure service which provides raw virtual machine instances, storage, and computation at pay-as-you-go utility pricing, and is currently the leading provider in this category. Developers are the typical target of this kind of Cloud Computing services.

B. Virtualization Technology

Virtualization technology [8, 9] is the key delivery technology in Cloud Computing. In a Cloud, virtualization refers primarily to platform virtualization, or the abstraction of physical IT resources from the users and applications using them. Virtualization allows servers, storage devices, and other hardware to be treated as a pool of resources rather than discrete systems, so that these resources can be allocated on demand.

Platform virtualization is performed on a given hardware platform by a control software, called a hypervisor or virtual machine monitor. This software creates a simulated computer environment, called a virtual machine, for its guest software. The guest software, which is often itself a complete operating system, runs just as if it were installed on a stand-alone hardware platform. The current leading virtualization and software providers include VMware [4], Xen [5], KVM [6], Force.com [10] and Microsoft Virtualization [11].

As a means of encapsulation of physical resources, virtualization solves several core challenges of datacenter managers and delivers specific advantages, including:

- ***Higher Utilization Rates***: Through virtualization, workloads can be encapsulated and transferred to idle or underused systems. This means that existing systems can be consolidated, so purchasing additional server capacity can be delayed or avoided.
- ***Resource Consolidation***: Virtualization allows for consolidation of multiple IT resources. Beyond server and storage consolidation, virtualization provides an opportunity to consolidate the systems architecture, application infrastructure, data and databases, interfaces, networks, desktops, and even business processes, resulting in cost savings and greater efficiency.
- ***Lower Power Usage/Costs***: Using virtualization to consolidate makes it possible to cut total power consumption and save significant costs.
- ***Space Savings***: Server sprawl remains a serious problem in most datacenters, but datacenter expansion is not always an option, with expensive building costs and cooling costs. Virtualization can alleviate the strain by consolidating many virtual systems onto fewer physical systems.

III. Case Study of Scalability and Performance of Web Applications

In the following sections, we will briefly discuss the key points in scaling web applications. We then present the experimental results on the performance of a scalable web application.

A. Scalability of Applications

Scalability is critical to the success of many organizations currently involved in doing business on the web and in providing information that may suddenly become heavily demanded. Scalability is a measure of the ability of an application to expand to meet enterprise business needs.

In general, enterprise can scale a given application by adding more or larger resources when needed. Resources can be many things including servers, processors, storage, and networking bandwidth. Scalable applications are able to operate normally as they grow and can have more resources added at any time to service more customer demands. Applications that are not scalable may encounter performance and service availability problems as demand increases. These kinds of non-scalable applications may not be able to take advantage of more resources.

B. Scaling Indicators

Although different web applications may not perform in the same way, there are very common scaling points where resources become constrained. For example, it is very unlikely that a web application can grow to meet business needs on a single server. At some point in time, it is required to add more servers to meet increasing demands in order to meet service quality requirements. The same is generally true for storage, networking, and other scaling point.

It is essential to understand how a web application consumes resources, how it behaves under high load, and what happens for the potential scaling points of the entire system in order to maintain the desirable performance of the web application. One way to understand how the application performs under increased customer load is to investigate where the scaling points are, and to use capacity planning to measure load on these points to add more resources when they are needed. Well-designed web applications allow the system to add more capacity to each scaling point as customer demand increases, thus growing the capacity to exactly meet increasing demand.

In order to scale the application in dynamic manner, it is common to use scaling indicators at the scaling point to monitor and track the performance. For web applications, typical scaling indicators at the web server may include:

- Number of concurrent users.
- Number of active connections.
- Number of requests per second.
- Average response times per request.

Once a scaling indicator is selected for use to scale the application, samplings of the scaling indicator are collected in real-time, and statistics is calculated periodically. Based on the historical trends and predictions derived from the statistics of the scaling indicator, scaling rules can be defined to scale up or down the amount of web application instances.

C. Performance of a Scalable Web Application

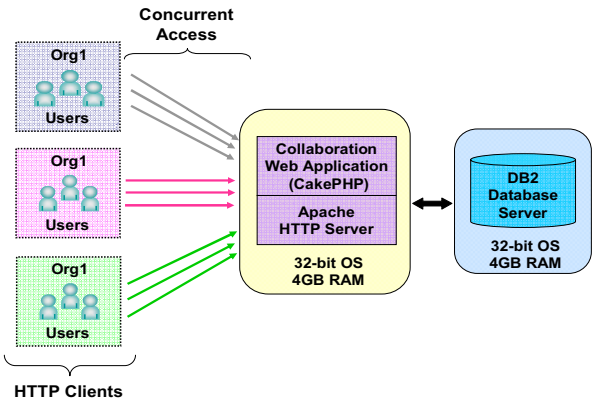

Figure 2. Experimental setup for studying the performance of a web application.

To explore the key scaling indicators, we have carried out the performance measurements on an online collaboration web application as illustrated in Fig. 2. This web application is intended for different groups of enterprise users to share online business documents and to organize and

track their organizational user contact information. It is developed based-on an open-source rapid development the CakePHP framework [12], and is deployed on an Apache HTTP Server [13] in a 32-bit operating system with 4GB of RAM's. The application also utilizes a remote back-end IBM DB2 relational database server to store all information including user profiles and credentials. In order to simulate concurrent users accessing the web application, a number of custom HTTP clients with different sets of user credentials are used to connect and to log on to the web application.

Our performance measurements are mainly focused on monitoring the usages in system resources and the access success/failure rates with a large number of users simultaneously accessing the web application. The following procedure was used to track and follow different user sessions for experimental measurements. The user session cookies returned by the web server upon individual logons are first cached in the HTTP clients. The cached session cookies corresponding to each logon users are then used subsequently to access different web pages in the web application.

Fig. 3 summarizes the failure rate of accessing a secure web page for 100 randomly selected users among all the logon users at every 50 logon-user intervals. Here, an access failure is defined as when an expected web page with secure information fails to return to the request HTTP client. From the results, it is observed that the failure rate starts to show up when the number of logon users reaches approximately 5,000 users, indicating that a single web application installed in an operating system can only support up to a threshold number of concurrent users (~5,000) at a given time. The starting failure point is found to correspond to the case where the total memory usage of the system reaches a maximum value and starts to reset and recycle as illustrated in Fig. 4 as a function of page hits. Here, we denote a page hit as a HTTP request to the web application which returns a complete web page (either a logon-success home page or an information-access page). The linear increase in system memory usage with increasing page hits certainly implies the existence of memory leaks in our web application system.

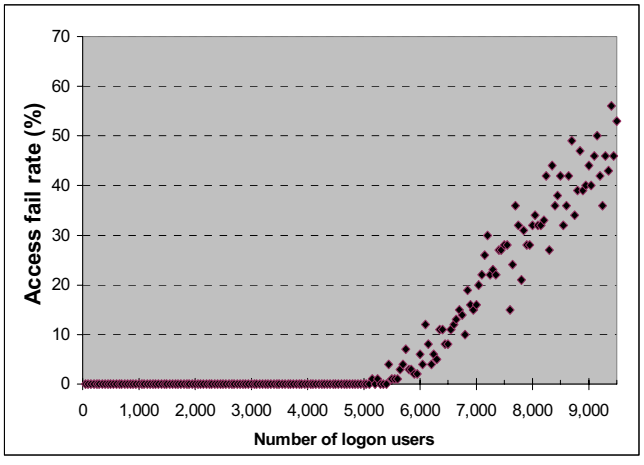

Figure 3. Access fail rate as a function of logon users.

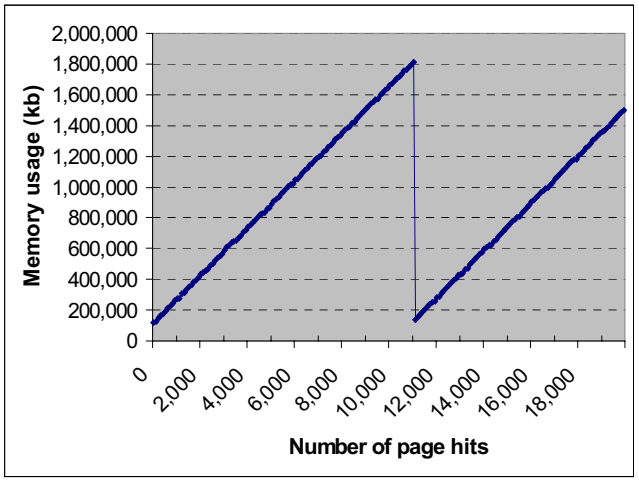

Figure 4. Total system memory usage as a function of page hits.

To support additional logon users beyond the observed threshold, it is required to deploy multiple copies of web applications in separate server nodes with the help of a load-balancer. This static load-balancing approach typically involves a pre-allocation of fixed number of server systems to meet anticipated capacities, and thus usually results in inefficient utilization of computing resources.

IV. Scaling Web Applications in a Cloud

In the following sections, we will present our architecture design for a novel scaling scenario, and the dynamic scaling algorithm for scaling a web application installed in virtual machine on a Cloud. The scenario is based on using a front-end load balancer to dynamically route user requests to back-end web servers that host the web application. The number of web servers should automatically scale according to the threshold on the number of current active sessions in each web server instance to maintain service quality requirements.

A. Architecture Design

We consider a scenario to provide quality of services on a given web application over the Internet at any time since there may have potentially unlimited number of users that could access the application unpredictably. This type of workload usually demands a short response time and high level of reliability and availability from the application. Thus, the web application should be made available without downtime and should provide the fastest response time regardless the amount of concurrent users accessing the system.

The main problem with such web applications is the inability to plan ahead or even predict the amount of users that will be accessing. A solution is to scale the web application in a dynamic manner and let the amount of web servers and web application components to grow (or shrink) on demand. A scalable architecture which effectively deals with this scenario has been given previously in [14], and is illustrated in Fig. 5 using a virtualized Cloud Computing environment. The architecture design includes a front-end load-balancer, a number of web app virtual machines, a

Provisioning sub-system, and a Service Monitor sub-system with a dynamic scaling algorithm.

Basically, the front-end load-balancer is an Apache HTTP Load Balancer which serves as a single point of entry for web requests and is used to route the requests to web servers that host the identical web applications. The web applications are deployed in Apache HTTP servers that are installed in virtual machines. These virtual machines are provisioned and started on-demand by a Provisioning sub-system in the Cloud infrastructure.

To control the action of provisioning or de-provisioning web server virtual machine instances, a dynamic scaling algorithm based on relevant threshold or scaling indicator of the web application is developed. The scaling indicator that we select here is the number of active sessions or logon sessions in each web application. This scaling indicator is found to be critical to the performance of our web application deployed on an Apache HTTP server according to our study results of performance and scalability as given in a previous section.

B. Load-Balancer and Web-App Virtual Appliances

Use of a load balancer such as Apache HTTP Load-Balancer allows incoming HTTP request to be routed into web servers that host the web applications. Since the Apache HTTP Load-Balancer configuration can be updated while the system is running, this permits the system to automatically and dynamically add new balancer members corresponding to new web server instances. The additional web servers enable the system to scale and thus allow to support more users with good response time for incoming HTTP requests. Moreover, to ensure that subsequent requests for a given user session are routed to the same web server instance for sticky session, a "*sticky_session*" cookie unique to each web application instance is generated and its value is attached to the "route" attribute of the corresponding balancer member in the Load-Balancer configuration file.

To simplify provisioning of additional resources, the web application and a corresponding web server is pre-installed and packaged in a virtual machine appliance image template which is then made available in an image repository of the Cloud. An appliance image is nothing more than the disk representation of a virtual machine pre-installed with an operating system and application software. An appliance template consists of two files: the configuration file, and the actual disk image. The configuration file represents the metadata about location of the disk file, display name, attached network and peripheral devices. Cloud Computing uses images as the building blocks for provisioning. Upon system command, new instances of appliances can be readily provisioned and started on-demand from the corresponding appliance image template.

C. Service Monitor and Provisioning Sub-system

The Service Monitor sub-system is responsible in gathering individual scaling indicators from the web applications and then calculating their moving average. In our current design, numbers of active sessions from individual web applications are used as the scaling indicators. A monitoring agent is installed in each web application to track the number of active sessions and to forward the number periodically to the Service Monitor sub-system.

Based on the moving average of the scaling indicator, a dynamic scaling algorithm, to be given in next section, is used to trigger a scaling event to the Provisioning sub-system. Depending on the updated statistics, action to scale up or down may be initiated. Scale up or down means that an event will be triggered instructing the Provisioning sub-system to start or shutdown web-server virtual machine instances running in the cloud. In case of scaling up the web application, the newly started virtual machine instance will run the web application. Once the web application instances are ready, the front-end load-balancer configuration file is then updated and refreshed to place them into active services.

The Provisioning sub-system is basically constructed from the IBM Tivoli Provisioning Manager (TPM) software product as given in [15]. This TPM software component can be used in general to execute manual tasks of provisioning and configuring servers, operating systems, middleware, applications, storages and network devices in the server clusters. It also provides a workflow execution framework to run Jython scripts and Java programs for workflow automation. To enable provisioning and management of networked servers and virtual machines in the Cloud, a number of TPM automation workflows have been developed for different operating systems and virtualization technologies. These workflows are then exposed from the TPM software tool as web services APIs ready to be used by other components in the Cloud.

D. Image-Based Provisioning

Image-based provisioning is a deployment and activation mechanism that clones a "golden" image template to create new virtual machine instances. One of the challenges with cloning virtual images is the handling of operating system, network, and application specific customization. Automating the provisioning of new virtual machines from a "golden" image template [16] can be accomplished by adding automation capabilities into the template image, combined with external automation scripts that control the deployment.

In our work, we perform the automated image-based provisioning of Linux-based virtual appliances based-on Xen hypervisor technology [5] in Red Hat Enterprise Linux system [17]. We utilize a simplified provisioning process that leverages the Linux disk image mount utility on the cloned Xen image and perform host name and network fix-ups before booting. The whole process can be summarized in two simple phases.

The first phase is to copy the "golden" image appliance template to the target Linux host system, mount the image file to the host system, and to perform fix-ups of hostname and network in the mounted file system. The fix-ups involve the first step of modifying the */etc/hosts* file, the */etc/sysconfig/network* file, and the */etc/sysconfig/network-scripts/ifcfg-ethx* files with correct entries for the new hostname and network settings. The second step of the fix-

ups is to copy the RSA public key of the host system into */root/.ssh/authorized_keys* file of the virtual machine. The former step will allow new virtual machine to boot with new host name and static IP address, thus allowing external network access. The latter step will allow the host system to remotely execute secure shell scripts in the virtual machine without the need to supply password. The second phase of the provisioning process involves the powering up of the virtual machine, and the remote execution of secure shell scripts on the VM for password change and for appliance-specific customization.

E. Dynamic Scaling Algorithm

As mentioned previously, the scaling algorithm is implemented in the Service Monitor sub-system, and is used to control and trigger the scale-up or down in the Provisioning sub-system on the number of virtual machine instances based on the statistics of the scaling indicator. A hybrid approach is used to support both goals of resource maximization on individual virtual machine instances and minimization of total number of instances, in contrast to the typical approach of load balancing among available resources.

The dynamic scaling algorithm, as given in Fig. 5 in pseudo procedures, is based on the scaling indicator Ai in each virtual machine instance in the Cloud. For the sake of illustration, we choose a scaling indicator in our implementation that corresponds to the number of active sessions in the web application of each instance. The algorithm first determines the current web application instances with active sessions above or below the given threshold numbers. If all instances have active sessions above the given upper threshold, a new web application instance will be provisioned, started, and then added to the front-end load-balancer. If there are instances with active sessions below a given lower threshold and with at least one instance that has no active session, the idle instance will be shutdown and destroyed from the system and its corresponding entry will be removed from the balancer members of the load-balancer. In each case, the load factors for all active instances will be re-calculated and then applied to the load-balancer to appropriately re-distribute the future request workloads to each instances.

Fig. 6 presents the experimental results on both the access failure rate and the access response time as a function of logon users with the implementation of our dynamic scaling algorithm on a Cloud. It is observed that there is no access failure (i.e. zero failure rate) even with more than 50,000 logon users to the web application (i.e. 10x more logon users than a single web application instance can support), and the access response times are kept relatively small and constant throughout the experiments. At one time under the peak load conditions, it is found that there are more than 12 virtual machine instances dynamically created and started in the Cloud.

For an instance **i** in $\mathbf{N_{Instance}}$

  If ($\mathbf{A_i/S_{Max}}$ >= $\mathbf{T_{Upper}}$) then

    Increment $\mathbf{N_{Exceed}}$

  If ($\mathbf{A_i/S_{Max}}$ < $\mathbf{T_{Lower}}$) then

    Increment $\mathbf{N_{Below}}$

  Record and sort all indexes **J** in ascending of $\mathbf{A_i/S_{Max}}$

If ($\mathbf{N_{Exceed}}$ == $\mathbf{N_{Instance}}$) then

  Provision and start a new instance

  Increment number of instances: $\mathbf{N_{Instance}}$

  Add new instance to Load-Balancer

If ($\mathbf{N_{Below}}$ >= 2) then

  Set **m** equal first index in **J**

  If ($\mathbf{A_m}$ == 0) then

    Remove instance m from Load-Balancer

    Shutdown instance m

    Decrement number of instances: $\mathbf{N_{Instance}}$

    Decrement $\mathbf{N_{Below}}$ and remove index m from **J**

    If ($\mathbf{N_{Below}}$ >= 2) then

      Set **n** equal first index in **J**

      Remove instance **n** temporary

      Decrement number of instances: $\mathbf{N_{Instance}}$

      Set normalized load factor $\mathbf{L_n}$ = 0

For an instance **i** in $\mathbf{N_{Instance}}$

  Evaluate normalized load factor:

  $\mathbf{L_i} = (1-\mathbf{A_i/S_{Max}}) / \mathbf{Sum}\ k=1,\mathbf{N_{instance}}\ [1-\mathbf{A_k/S_{Max}}]$

  Apply new load factors L_i to Load Balancer

where

$\mathbf{A_i}$: Number of active sessions in instance **i**

$\mathbf{S_{Max}}$: Maximum sessions per instance (e.g. 5,000)

$\mathbf{T_{upper}}$: Session upper-threshold (e.g. 80%)

$\mathbf{T_{Lower}}$: Session lower-threshold (e.g. 20%)

$\mathbf{N_{Instance}}$: Number of existing instances

$\mathbf{N_{Exceed}}$: Number of instances exceeding session upper-threshold

$\mathbf{N_{Below}}$: Number of instances below session lower-threshold

Figure 5. Dynamic Scaling Algorithm for Virtual Appliance Instances in a Cloud.

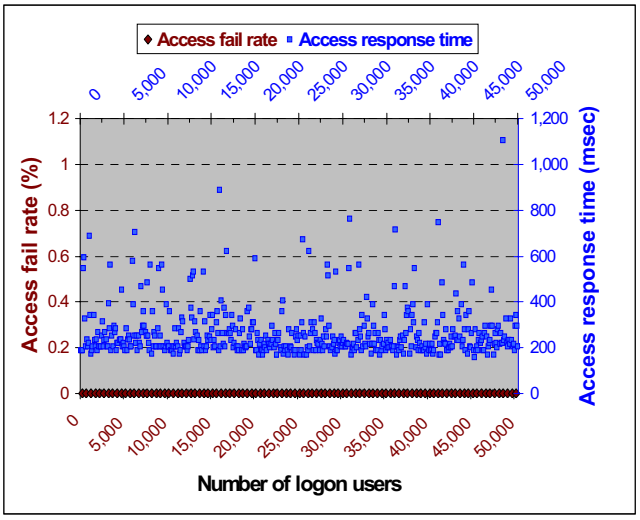

Figure 6. Experimental results on access fail rate and access response time as a function of logon users with the implementation of the dynamic scaling algorithm on a Cloud.

V. Related Work

In the last decade, the goals of distributed systems have been centered on the decoupling of interfaces from implementation, the hosting models, subscription-based computing, the service orientation architecture (SOA) [18], and social collaboration. Recently, Internet-hosted distributed, multi-tenant [19] applications with connectivity to internal business applications, referred as software as a service (SaaS) [6], are gaining popularity. Businesses are leveraging datacenters hosted by third-party providers with Internet-accessible, virtualized Cloud Computing environment to alleviate concerns about hardware, software, maintenance, availability, reliability, and scalability.

Particularly, scalability has been one of the principal topics discussed regarding Cloud Computing. Scalability is the ability of an application to be scaled up to meet demand through replication and distribution of requests across a pool or farm of servers. Dynamic scalability of web applications in virtualized Cloud Computing based on a hybrid approach of maximizing resource utilization and minimizing resource allocation as outlined in the present work has not been much discussed. Most of the previous works [20, 21, 22] on web scalability have been reported through implementation of static load balancing solutions with server clusters.

VI. Conclusion

In summary, we have presented a novel architecture design and a dynamic scaling scenario to investigate the scalability and performance of web applications on a Cloud Computing environment. The system is constructed with a front-end load balancer to route and balance user requests to web applications deployed on web servers in virtual machine instances in the Cloud. A dynamic scaling algorithm for automated provisioning of the virtual machine resources based on number of active sessions has also been introduced. Our algorithm is aimed to maximize resource utilization in each virtual instance while minimizing the total number of deployed instances.

Our work has demonstrated the compelling benefits of the Cloud which is capable of handling sudden load surges, and delivering IT resources on-demands to users in a better and cheaper way. The automated provisioning, dynamic allocation and rapid scaling capabilities of the Cloud are the essential elements in providing higher resource utilization, thus reducing infrastructure and management costs.

References

[1] G. Gruman, "What cloud computing really means", InfoWorld, Jan. 2009.

[2] R. Buyya, Y. S. Chee, and V. Srikumar, "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities", Department of Computer Science and Software Engineering, University of Melbourne, Australia, July 2008, pp. 9.

[3] D. Chappell, "A Short Introduction to Cloud Platforms", David Chappell & Associates, August 2008.

[4] VMware ESX Server, VMware Inc., http://www.vmware.com/products/vi/esx/

[5] Xen Hypervisor, http://www.xen.org/

[6] Kernel-based Virtual Machine (KVM), http://www.linux-kvm.org/page/Main_Page

[7] E. Knorr, "Software as a service: The next big thing", InfoWorld, March 2006.

[8] Virtualization Technology, http://www.kernelthread.com/publications/virtualization/

[9] VMware Inc., "Understanding Full Virtualization, Paravirtualization, and Hardware Assist", VMware, 2007, http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf.

[10] Force.com, http://www.salesforce.com/platform/

[11] Microsoft Virtualization, http://www.microsoft.com/virtualization/

[12] CakePHP, http://cakephp.org/

[13] Apache HTTP Server, http://httpd.apache.org/

[14] T. Chieu, A. Mohindra, A. Karve, and A. Segal, "Dynamic Scaling of Web Applications in a Virtualized Cloud Computing Environment", Proceedings of the 2009 IEEE International Conference on e-Business Engineering, Macau, China, Oct. 2009, pp. 281-286.

[15] IBM Tivoli Provisioning Manager Products, http://www.ibm.com/software/tivoli/products/prov-mgr/

[16] L. He, S. Smith, R. Willenborg and Q. Wang, "Automating deployment and activation of virtual images", IBM developerWorks, Aug. 2007, http://www.ibm.com/developerworks/websphere/techjournal/0708_he/0708_he.html

[17] Red Hat Enterprise Linux, http://www.redhat.com/rhel/

[18] T. Erl, "Service-oriented Architecture: Concepts, Technology, and Design", Upper Saddle River, Prentice Hall, 2005.

[19] F. Chong, G. Carraro, and R. Wolter, "Multi-Tenant Data Architecture", Microsoft Corporation, 2006.

[20] V. Ungureanu, B. Melamed, and M.Katehakis, "Effective Load Balancing for Cluster-Based Servers Employing Job Preemption", Performance Evaluation, 65(8), July 2008, pp. 606-622.

[21] L. Aversa and A. Bestavros. "Load Balancing a Cluster of Web Servers using Distributed Packet Rewriting", Proceedings of the 19th IEEE International Performance, Computing, and Communication Conference, Phoenix, AZ, Feb. 2000, pp. 24-29.

[22] V. Cardellini, M. Colajanni, P. S. Yu, "Dynamic Load Balancing on Web-Server Systems", IEEE Internet Computing, Vol. 33, May-June 1999, pp. 28 -39.