SaaS Performance and Scalability Evaluation in Clouds

Jerry Gao①②, Pushkala Pattabhiraman, ② San Jose State University, USA
Xiaoying Bai ②Tsinghua University, China
W. T. Tsai Arizona State University, USA

Abstract: Cloud computing not only changes today's computing infrastructure, but also alters the way of obtaining computing resources, managing and delivering software and services. Meanwhile, cloud computing brings new issues, challenges, and needs in performance testing, evaluation and scalability measurement due to the special features of cloud computing, such as elasticity and scalability. This paper focuses on performance evaluation and scalability measurement issue for Software as a Service (SaaS) in clouds. It proposes new formal graphic models and metrics to evaluate SaaS performance and analyze system scalability in clouds. In addition, the paper reports an evaluation approach based on Amazon's EC2 cloud technology and detailed case study results using the proposed models and metrics.

KEYWORDS: SaaS performance evaluation and scalability measurement, cloud performance evaluation, scalability modeling, cloud scalability measurement

1. Introduction

Cloud computing not only changes today's computing infrastructure, but also alters the way of obtaining computing resources, managing and delivering software, technologies and solutions. Meanwhile, cloud computing also brings new issues, challenges, and needs in cloud-based application testing and evaluation. One of them is performance evaluation and scalability measurement for online applications which are provided as a service in cloud infrastructures. The major causes are the special cloud features, such as elasticity and scalability.

In the past decades, there were numerous published technical papers focusing on scalability analysis and performance evaluation. As discussed in Section 5, most of published papers address performance validation and scalability evaluation issues and needs in conventional distributed and parallel systems instead of cloud-based applications [1]-[11][26]. Hence, these research results have limitations to support performance evaluation and scalability analysis for cloud-based applications (such as SaaS applications) due to the following distinct features of SaaS in clouds.

- Cloud-based scalable operation system environment with diverse provisional computing resources – All SaaS in a cloud environment are operated and supported by a scalable cloud infrastructure with both physical and virtual computing resources, which are provisioned statically or dynamically.
- Scalability – Providing ***elastic*** scalability for SaaS and cloud-based applications in a cloud has been claimed to be a major advantage and a distinct feature in cloud computing. How to validate and assure this claim becomes one important focus in service quality validation and evaluation.
- Utility service billing and pricing model – Each SaaS provides its customers with a pre-defined pricing model so that each of them is charged based on received services as a utility billing.
- Service-Level-Agreement (SLA) – Unlike the conventional software services, cloud-based applications and SaaS must deliver the provided services to clients and customers based on a pre-defined and posted SLA to assure system service quality. One of the focuses in SLA is to assure the delivery of the required performance and scalability for customers in a cloud.

In the last decades, many published papers discussed the issues, models, metrics, and tools for evaluating system performance and scalability in a distributed and parallel environment [1]-[11]. There are three groups of indicators about system performance evaluation. They are listed below.

- ***Computing resource indicators***: This group of indicators relate to computing hardware and software resource allocation and utilization in CPU, disk, memory, networks, and other system resources.
- ***Workload indicators:*** These indicators include the number of currently online access users and clients, application service loads (such as throughputs), and connectivity traffic loads.
- ***Performance indicators:*** This group of indicators includes application processing speed, system reliability, availability, and scalability based on the given QoS standards.

Although there are a number of published papers addressing the issues and challenges in cloud testing and testing as a service (TaaS) [26][27][28][29], very few of them focus on SaaS performance and scalability evaluation in clouds.

This paper focuses on the issue and challenge in cloud-based system performance evaluation and scalability analysis. It first discusses basic concept, major objectives and focuses in performance evaluation and scalability analysis for SaaS in clouds. Meanwhile, it highlights the new challenges and related needs. The major technical contributions of this paper include two folds.

- The paper proposes formal and graphic models and related metrics for evaluating SaaS performance and scalability in clouds. They can be useful to develop virtualization solutions to monitor and analyze SaaS system performance and scalability in clouds.
- The paper presents a cloud-based evaluation approach and reports Amazon EC2-based case study results using the proposed models and metrics.

The rest of the paper is structured as follows. The next section discusses basic understanding about SaaS performance evaluation and scalability measurement in clouds. It includes basic concepts, objectives and focuses, issues and needs. Section 3 proposes SaaS-oriented performance and scalability measurement models and related metrics. Section 4 presents our case study in Amazon EC2 cloud environment and related results. It presents a SaaS performance evaluation approach, cloud APIs and EC2 cloud environment, and reports our case study results. In addition, Section 5 reviews and discusses the published papers and related research work on this subject. Finally, the conclusion remarks and future research directions are given in Section 6.

2. Understanding System Performance and Scalability Evaluation in Clouds

Before addressing the issues and challenges of performance evaluation and scalability analysis in clouds, we must have some basic understanding about related concepts, such as objectives, focuses, tasks in clouds. In [23], Chapter 11 discusses the basics of performance testing and evaluation for components and systems. It covers the primary focuses, models, and metrics. Table 1 summarizes these focuses in performance validation and evaluation. They are useful indicators for validating the performance scalability of SaaS applications in clouds.

Table 1. Performance Evaluation and Validation Focuses

Targeted Focuses	Detailed Descriptions
Resource Utilization Indicators	CPU, Memory, Cache, Disk Storage, Network Traffic, such as Network-In and Network-Out
Performance Indicators	Process speed, system-user-response time, task speed, transaction speed, latency, and etc.
	Reliability, availability, throughput, and scalability

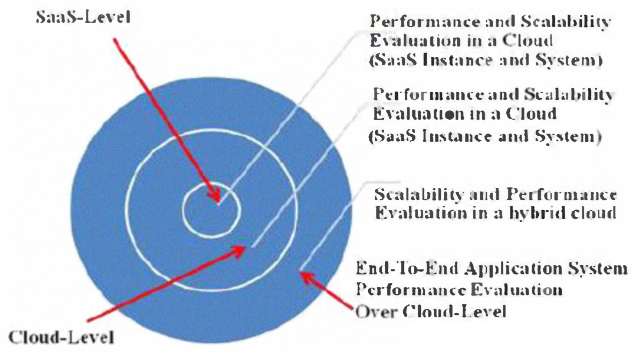

SaaS-Level
Performance and Scalability Evaluation in a Cloud (SaaS Instance and System)
Performance and Scalability Evaluation in a Cloud (SaaS Instance and System)
Scalability and Performance Evaluation in a hybrid cloud
End-To-End Application System Performance Evaluation Over Cloud-Level
Cloud-Level

Figure 1. Scalability and Performance Evaluation in Clouds

SaaS performance and scalability can be evaluated and measured in a cloud-based environment at three different levels as shown in Figure 1. Figure 2 displays three different testing environments at the three levels.

- **At SaaS Level** – An individual SaaS application (or instance) is evaluated and validated in a private (or public) cloud for its system performance and scalability.
- ***On Cloud Level*** – Multiple SaaS instances and cloud-based applications in a private (or public) cloud are evaluated and validated for performance and scalability.
- ***Over Cloud Level*** – Cloud-based applications over a hybrid cloud infrastructure are evaluated and validated for end-to-end system performance and scalability.

There are four primary objectives in evaluating SaaS applications to assure the quality of services. They are listed below.

- Check and measure the resource allocation and utilization for SaaS applications in a cloud environment to assure the quality delivery of cloud-based computing resources.
- Check and measure different performance indicators of SaaS applications in a cloud infrastructure to find out the performance issues and bottlenecks.
- Discover and measure application system capacity, such as system load boundaries and system capacity thresholds.
- Evaluate the scalability of SaaS applications in a cloud infrastructure the different levels to assure its elastic scalability.

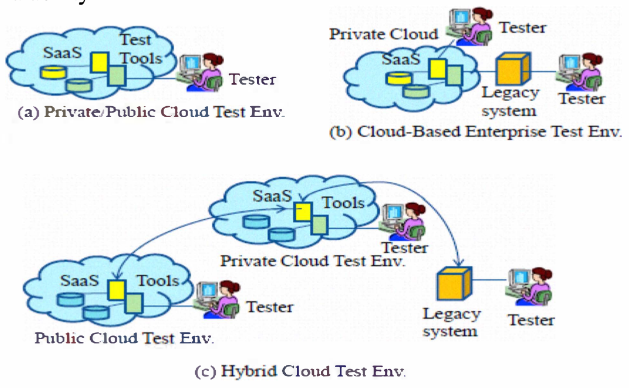


Figure 2 Different Test Environments in Cloud

There are several distinct differences in performance evaluation and scalability analysis between conventional software and cloud-based applications. The ***first*** is the underlying evaluation system environment since a cloud infrastructure provides an elastically scalable computing infrastructure with shared computing resources that are auto-provisioned and de-provisioned. The other two differences are listed below.

- ***SLA-based evaluation*** – All SaaS and cloud-based applications must provide a pre-defined service-level agreement (SLA) as a contract to clients and customers concerning the quality of services. Since the quality requirements about system performance and scalability must be addressed as a part of a SLA, they must be validated and measured based on the contracted SLA. Current loud vendors provide a pre-defined SLA to their clients for their provided cloud infrastructure and service software. Similarly, SaaS vendors also provide a SLA to clients for their offered applications in a cloud.

- ***Utility based evaluation*** – Utility billing is one of the major service delivery features in cloud computing. This implies that the delivery of diverse services of cloud-based applications (such as SaaS) must be charged in a pre-defined price model. Elastic scalability of clouds brings a new challenge task in performance evaluation scalability analysis. That is how to use a transparent approach to monitor and evaluate the correctness of the utility bill based on a posted price model during system performance evaluation and scalability measurement.

Table 2 SaaS/Cloud Scalability Evaluation Focuses

Targeted Focuses	Detailed Descriptions
Scaling-up Capability and Factor	Validating and evaluating SaaS applications to see how well a cloud is capable to scale-up its capacity to accommodate its increasing system loads with automatic provisioning of computing resources.
Scaling-out Capability and Factor	Identifying and discovering the scaling-out capability and factor of SaaS applications in a cloud when the system load and resource usage are increased.
Scaling-down Capability and Factor	Identifying and discovering the scaling-down indicator of SaaS applications in a cloud environment when the system load and resource usage are decreased.
Scalability Cost	Validating the costs of dynamic scalable system resource utilization and services based on the economic scales in

	the given SLA.

Table 2 highlights typical focuses in evaluating and validating SaaS performance and scalability in (on or over) clouds. Figure 2 presents four different types of needs at the three levels.

- Resource utilization and allocation measurement for underlying computing resources.
- Performance measurement under the allocated computing resources for diverse system loads.
- Scalability measurement in different cloud infrastructures.
- Cost-driven evaluation based on a pre-defined price model.

To address these needs, engineers need proper evaluation models and computing approaches (such as metrics and algorithms). This paper addresses these needs for SaaS applications in a cloud.

3. Evaluation Models and Metrics

Although some current cloud technologies offer certain cloud resource monitor tools with APIs (such as *CloudWatch* in Amazon EC2) to allow cloud clients to collect some indicators about the usage of cloud computing resources, there is a lack of analytic models supporting dynamic evaluation and analysis of application system performance and scalability. As discussed in Section 5, existing research results focus on performance evaluation and scalability measurement issues and metrics in conventional distributed (or parallel) systems using heuristic approaches. They did not address the need of an analytic approach to support the dynamic and virtualized monitoring and analysis of utility-based performance evaluation and scalability analysis for SaaS applications in clouds based on the given service-level-agreement.

This paper presents such analytic models in a graphic format (as a radar chart) as a base to develop well-defined metrics that enable dynamic visualization and evaluation for system performance and scalability of SaaS applications in a cloud. A **radar chart** is known as a web chart, irregular polygon, or spider chart [25]. It is a graphical method of displaying multivariate data in the form of a two-dimensional chart of three or more quantitative variables represented on axes starting from the same point. The relative position and angle of the axes is typically uninformative. In this paper, we use a radar chart as a graphic base to develop our formal models and metrics to support the analysis, monitoring, and evaluation of numerous quantitative performance and scalability parameters and indicators.

3.1. Measuring System Resources Allocation and Utilization

Monitoring system resource allocation and utilization is one important task in system performance testing and evaluation. Due to the elasticity of a cloud, diverse computing resources in a cloud are automatically provisioned to support the execution of SaaS applications (or instances) in a cloud based on system performance and loads as well as scalability configurations.

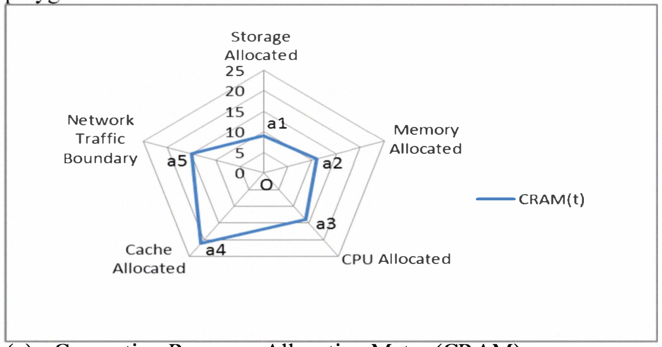
Since computing resource usage in a cloud usually is charged by the cloud vendor based on the posted pricing model, there is strong interest and need from SaaS vendors to find out a cost-effective way to validate the correctness and effectiveness of computing resource allocation and utilization in clouds to provide a reliable third-party view and reference. Here we use a polygon-based graphic model as a base to define a generic resource allocation meter and utilization meter to evaluate SaaS applications in clouds.

Uniform Resource Allocation Meter

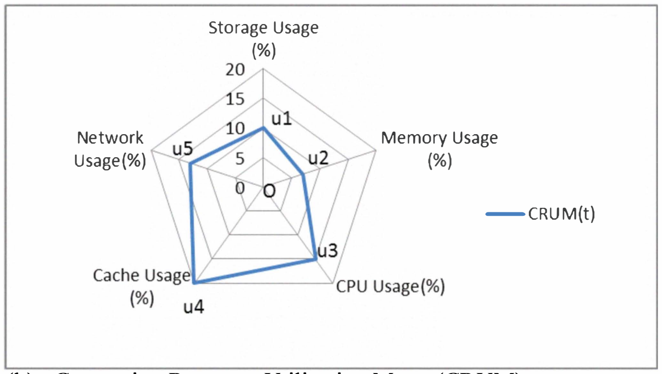
For each given SaaS (say S) in a cloud, we use an irregular polygon (presented as a ***radar chart)*** as a base to represent its Computing Resource Allocation Meter, denoted as ***CRAM(S, t),*** which can be used to an indicator to present the total amount of resource allocations for S in a cloud at the evaluating time ***t***. In this ***radar chart***, each axe is used as an indicator to present one type of allocated computing resources for S in a cloud. Hence the total computing resources of S at time t can be presented and viewed as the total area of its corresponding irregular polygon in this chart. To compute the read of this polygon, we only need to compute the area of each related sub-polygon (that is a triangle of three points O, a_i, and a_{i+1}), and then add them together. The detailed computation formula is listed below.

$$CRAM(S,t) = 0.5 * \sin(2\pi / n) * \sum_{i=1}^{n} a_i * a_{i+1} \qquad \textit{(1)}$$

Where $\alpha_1,\ldots, \alpha_n$ represent the diverse amounts of different types of allocated computing resources to S by a cloud at the evaluating time ***t***. ***CRAM(S, t)*** can be used as a dynamic resource allocation meter for SaaS applications in a cloud during performance validation and evaluation. Figure 3(a) shows an example that depicts five types of allocated computing resources for a deployed SaaS application in a cloud, including CPU, Cache, memory, network usage (such as data network-in and data network-out), and disk storage. Since each type of computing resources has its own allocation units, its corresponding axe in the radar chart must have its own presentation unit and scales, for example MB and GB for memory. The blue polygon in Figure 3(a) represents the total computing resources allocated to ***S*** at the evaluation time ***t***. The formula given in (1) shows the way to compute the area of this polygon.



(a) Computing Resource Allocation Meter (CRAM)



(b) Computing Resource Utilization Meter (CRUM)

Figure 3.Uniform Resource Allocation Meter and Utilization Meter

Uniform Resource Allocation Meter

SaaS vendors and application clients in a cloud must interested in finding out the actual computing resource usage during system services. Hence, a computing resource utilization meter is a useful virtualization tool to present and measure the cloud resource utilization of any given SaaS application during system validation in a cloud. Similarly, we can easily define the resource utilization meter for S below.

For each given SaaS (say S) in a cloud, we use a radar chart as its Computing Resource Utilization Meter, denoted as ***CRAM(S, t)*** to present its system utilization of selected computing resources at the evaluation time ***t***. In this radar chart, each axe is used a utilization indicator for one type of computing resources of a deployed SaaS application in a cloud. Similarly to Formula (1), the following formula is given to assist the computation of its total system resource utilization,

$$CRUM(S,t) = 0.5 * \sin(2\pi / n) * \sum_{i=1}^{n} u_i * u_{i+1} \qquad (2)$$

Where $\mu_1,\ldots,\mu_n$ represent different computing resource utilization of S in a cloud at the evaluating time t. *CRUM(S, t)* can be used as a dynamic resource utilization meter for S in a cloud to assist engineers and clients in system performance evaluation and monitoring. Figure 3(b) depicts five types of computing resources used by S in a cloud. The blue polygon here presents a snapshot picture of the total computing resource usage of diverse system resource at time ***t***. The given formula (2) can be used to compute the area of the blue polygon that represents the total computing resource usage for S.

3.2. Measuring System Performance in Clouds

Besides system resource utilization, it is common to collect and measure system performance indicators of SaaS in a cloud for performance testing and evaluation based on the given QoS and SLA. A typical performance parameter set includes the processing speed (such as user response time), system utilization, throughput, reliability, and availability. To support the diverse need of performance evaluation of different performance parameters, we use a radar chart as a ***System Performance Meter (SPM)*** to provide a virtualized view about the system performance of a deployed SaaS/application in a cloud. Unlike the existing work, this approach provides a comprehensive view about system performance in a dynamic way with a uniform format although different performance parameters may have their own scale units.

Let's use ***SPM(S, t)*** to denote the System Performance Meter of SaaS (S) in a cloud during the system performance evaluation at time ***t***. As shown in Figure 4, a snapshot of the total system performance for S is presented as a blue polygon (as a radar chart). Similar to formula (2), the area of can be formally defined and computed below.

$$SPM(S,t) = 0.5 * \sin(2\pi / n) * \sum_{i=1}^{n} p_i * p_{i+1} \qquad (3)$$

Where p_1, ..., p_n represent different performance indicators for S during performance evaluation at time ***t*** in a cloud.

Clearly, ***SPM(S, t)*** presents different polygons at the different time. Computing and analyzing these polygons provides an effective mean to compare and monitor system performance of SaaS applications in a cloud.

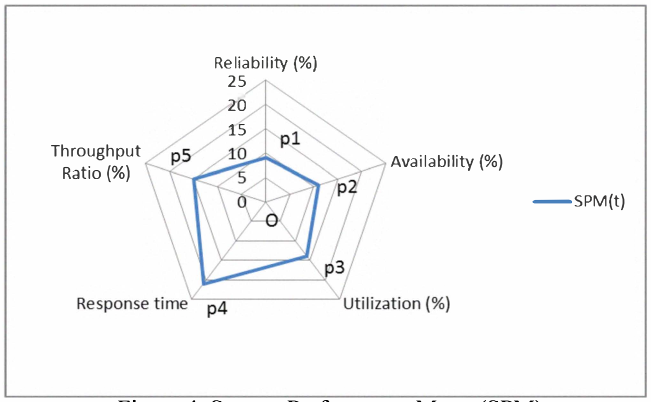


Figure 4. System Performance Meter (SPM)

3.3. Measuring System Capacity in Clouds

Measuring system capacity is an important task in system performance and scalability testing. Hence, discovering and evaluating SaaS system capacity becomes an important task in cloud-based system evaluation and testing. This is not only a part of system performance evaluation and testing, but also an important focus in SaaS and cloud scalability measurement. Unfortunately, there is a lack of existing research work addressing dynamic evaluation and analysis of system capacity and scalability in a cloud. Here, we use a radar chart as a base to develop a formal model and metrics to present system capacity and scalability of SaaS applications in a cloud.

System Load Meter (SLM)

A software system capacity can be defined as its capability to handle system loads based on the allocated system resources. Each SaaS application must have different types of system loads during system evaluation in a cloud. These system loads can be classified into the following classes.

- ***User access load***, which refers to the number of concurrent users who access the system in a given time unit.
- ***Communication traffic load***, which refers to the amount of incoming and outgoing communication messages and transactions in a given time unit during system performance and evaluation.
- ***Data storage access load***, which refers to the underlying system data store access load, such as the number of data store access, and data storage sizing.

To effectively present the system load for a deployed SaaS application in a cloud during performance evaluation, we use a simplified radar chart (an irregular triangle) as a model to present the dynamic system load in the above three types of system loads

Figure 5 shows an example, which represents the overall system load for a deployed SaaS in a cloud during system performance time ***t***. We need to understand that each type of system loads has its own scale unit. The red cloud polygon in Figure 5 presents the overall system load during the evaluation period ***t***. Clearly, the area of the red polygon represents the total system load during ***t***.

For a deployed SaaS application S in a cloud, its comprehensive ***System Load Meter***, denoted as ***SLM(t)*** can be used as an effective mean to evaluate diverse system loads during the system validation and evaluation time ***t***. When we only consider three types of system loads, there are only three axes in a radar chart. Hence, the area of the rad polygon centered at point O can be computed based on three small triangles. Since the angle between any two axes is 120°, therefore, ***SLM(t)*** can be computed below using the similar way in the previous formula.

$$SLM(t) = \sin 120° \times [UAL(t) \times CTL(t) + UAL(t) \times SDL(t) + CTL(t) \times SDL(t)] \quad (4)$$

Where,

- ***CTL(t)*** stands for the communication traffic load during ***t***.
- ***UAL(t)*** stands for the system user access load during ***t***, and
- ***SDL(t)*** stands for the system data load during ***t***.

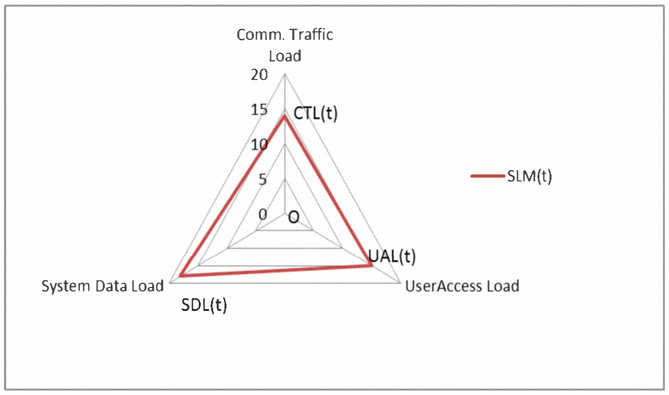


Figure 5. System Load Meter (SLM)

System Capacity Meter (SCM)

Evaluating system capacity of a deployed SaaS application becomes a necessary task to system scalability analysis in a cloud. Here, we use a radar chart as a graphic model to define the system capacity for SaaS applications in a cloud under one of its configurations based on the following three factors (as three axes):

a) the current system load,
b) the current allocated resources, and
c) the current system performance.

In a cloud, the system capacity for a SaaS application changes from to time due to the changes of the system loading and the allocated system resources. To achieve the elastic scalability in a cloud, a cloud use different deployed scalability solutions for auto-provision of computer resources to assure the agreed system performance requirements (in a SLA). Whenever, more system resources are allocated, the system performance will be improved, and system loads will be balanced.

Figure 6(a) use a radar chart as a mean to present a general example of the System Capacity for S in a cloud during the evaluation time t. In this chart, three axes are used to present three indicators: system performance (SP), System Load (SL), and Resource Allocated (AR). The blue polygon in Figure 6 represents the total system capacity at evaluation time ***t***.

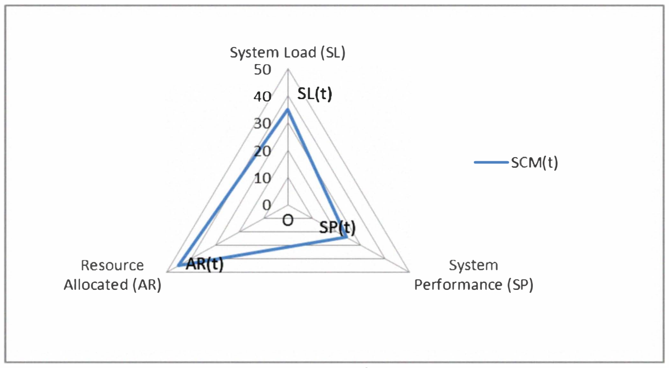
Let's use ***SCM(t)*** to stand for the system capacity meter for a SaaS (S) in a cloud at time t. ***SCM(t)*** presents its system capacity. Similar formula (4), the detailed formula to compute the system capacity is given below.

$$SCM(t) = \sin 120° \times [SL(t) \times SP(t) + SP(t) \times AR(t) + AR(t) \times SL(t)] \quad (5)$$

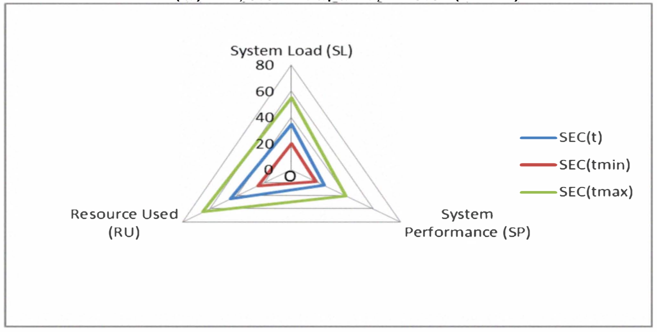
Where ***SL(t)***, ***SP(t)***, and ***AR(t)*** represent the data values of the system load, system performance, and allocated system resources respectively during system evaluation (or validation) time. As discussed before, these three parameters can easily evaluated based on ***SLM(t)***, ***SPM(t),*** and ***CRAM(t)***. With this model, we can easily compute, monitor, and measure dynamic system capacity of deployed SaaS (or applications) in a cloud for any given time period ***t***.

System Effective Capacity (SEC)

Clearly, the ***SCM(t)*** given in formula (5) provides a useful meter for SaaS vendors and cloud vendors since it presents the system capacity during the evaluation time. However, ***SCM(t)*** does not reflects the actual system capacity because it is based on the allocated computing resources instead of the actual used resources. Clearly, it is useful and practical for SaaS vendors and clients to find out the real effective system capacity based on the real utilization of the computing resources for a deployed SaaS (or an application) in a cloud. This need becomes obvious and critical due to the cost of allocated resources.



(a) System Capacity Meter (SLM)



(b) System Effective Capacity Meter (SEC)

Figure 6. System Capacity Meters

To find out the effective system capacity, we need to define another meter used to accurately represent the system effective capacity that delivers and supports the expected system performance according to the given SLA under the diverse system loads by effectively utilizing the allocated resources.

Figure 6(b) shows a radar chart based model presenting the effective system capacity (denoted as ***SEC(t)***) during the evaluation time period t for a deployed SaaS (or an application) in a cloud. ***SEC(t)*** can be evaluated based on three parameters: a) system load ***SL(t),*** b) system performance ***SP(t),*** and c) system resource utilization ***RU(t).***

Similar to formula (5), the system effective capacity can be analyzed by computing the area of the corresponding polygon. The details are listed below.

$$SEC(t) = \sin 120° \times [SL(t) \times SP(t) + SP(t) \times RU(t) + RU(t) \times SL(t)] \quad (6)$$

Where *SL(t), SP(t) and RU(t)* represent the following parameters respectively:

- The system load reading based on its SLM.
- The system performance reading based on its SPM.
- the resource utilization reading based on its RUM.

Figure 6(b) shows an example of ***SEC(t)*** based on three meters. Since each meter has its own minimum and maximum readings, we can easily define and compute the upper bound (denoted as $SEC(t_{max})$ and lower bound (denoted as $SEC(t_{min})$) for the system effective capacity during time ***t*** below.

$$SEC(t_{min}) = \sin 120° \times [SL(t_{min}) \times SP(t_{min}) + SP(t_{min}) \times RU(t_{min}) + RU(t_{min}) \times SL(t_{min})] \quad (7)$$

Where

- ***t*** represents the evaluation time period.
- $SL(t_{min})$ *represents* the minimum reading in system load meter ***(SLM)*** during time ***t.***
- $SP(t_{min})$ represents the minimum system performance reading based on its system performance meter ***(SPM)*** during time ***t***.
- $RU(t_{min})$ represents the minimum reading in computing resource utilization meter ***(CRUM)*** during time ***t***.

$$SEC(t_{max}) = \times [SL(t_{max}) \times SP(t_{max}) + SP(t_{max}) \times RU(t_{max}) + RU(t_{max}) \times SL(t_{max})] \quad (8)$$

Where

- ***t*** represents the evaluation time period.
- $SL(t_{max})$ *represents* the maximum reading in system load meter ***(SLM)*** based during time ***t.***
- $SP(t_{max})$ represents the maximum system performance reading based on its system performance meter ***(SPM)*** during time ***t***.
- $RU(t_{max})$ represents the maximum reading in computing resource utilization meter ***(CRUM)*** during time ***t***.

3.4. Measuring System Scalability in Clouds

According to wikipedia, "Scalability is the capability to increase resources to yield a linear (ideally) increases in service capacity. The key characteristic of a scalable application is that additional load only requires additional resources rather than extensive modification of the application itself." Here, we follow this concept, and formally define system scalability based on our proposed evaluation models and metrics for system loads, system performance, and system resource utilization below.

To effective present system scalability for SaaS applications in clouds, we define a system scalability meter based on system effective capacity during a given time period t.

Since for each SaaS application, its vendor must make sure that it delivers its minimum system performance requirements under diverse system loads based on its underlying computing resources in a scalable cloud environment.

As shown in Figure 8, for any given SaaS application instance, we formally define its Effective System Scalability (ESS) based on its Effective System Capacity (SEC) in (6) below. It refers to the ratio between the system load increase and system capacity increase.

$$ESS = [SL(t) - SL(min)]/[SEC(t) - SEC(min)] \quad (7)$$

Where SL(t) stands for the system load during the evaluation time period (t) and SL(min) stands for the minimum system load in the system load meter (SLM). SEC(t) represents the system effective capacity for S during the evaluation time period t, and SEC(min) represents the minimum effective system capacity for S.

Moreover, we can evaluate the ***Effective Scalable Range (ESR)*** for S based on the difference between the maximum and minimum effective system capacity.

$$ESR = SEC(max) - SEC(min) \quad (8)$$

4. Case-Study Results

To validate the proposed virtualizable models and metrics, we have conducted a case study using Amazon EC2. The major objectives of this study include the followings:

- Apply one modern cloud technology (Amazon EC2) and the related supporting APIs and tools to see what system performance parameters and scalability indicators of SaaS applications are collectable and traceable in a cloud.
- Check if the proposed models and metrics are useful and effective to evaluate SaaS performance and scalability in a cloud infrastructure (such as an Amazon EC2 cloud).
- Validate the effectiveness of the presented systematic approach in measuring the system performance and scalability for selected applications in an EC2 cloud.

4.1. The Used Cloud Infrastructure, Tools, and APIs

In this case study, we used the following cloud technology and related APIs and tools.

- ***Amazon EC2 Cloud APIs (see Table 3)***
  The Amazon EC2 cloud technology provides necessary cloud APIs and service tools to manage and monitor a set of resource utilization parameters and primitive system performance metrics at two different levels: a) the instance level, and b) the cloud level.

- ***CloudWatch***

This is an instance-level monitor service that enables cloud users to monitor the performance of the dynamically created EC2 instances by collecting performance parameters through the provided APIs. Table 4 lists the resource and performance parameters provided in CloudWatch to monitor EC2 instances. Only these parameters can be set up as scaling triggers (threshold points) in a cloud for scaling purposes.

- **AutoScaling**
  It offers the API service to enable cloud users to set up the scaling parameters for EC2 instances to enable scale-up and scale-down actions automatically. There are two important parameters. ***AutoScalingGroup*** is one of them. It refers to is a collection of EC2 instances on which the scalability conditions are applied. The other is ***scaling trigger*** that defines a scaling condition, for example, setting the average of CPU utilization to be no more than 70%.

Applying different loads onto the selected EC2 instances, users can collect the values of those parameters to evaluate the scaling facts of the selected EC2 instances. For example, users can use the provided API parameter (known as ElasticLoadBalance) to enable dynamic load balancing between EC2 instances for the SaaS application.

Table 3. Amazon EC2 Cloud APIs

APIs and Utility for Resources and Performance Monitor	URL
CloudWatch	aws.amazon.com/cloudwatch/
AutoScaling	aws.amazon.com/autoscaling/
LoadBalancing Cloud	aws.amazon.com/elasticloadbalancing/

Table 4. Resource & Performance Parameters from CloudWatch

Instance parameters	Load Balanced Instance Parameters	Statistical Rendition of Parameters
CPU Utilization	Latency	Minimum
Network In	RequestCount	Maximum
Network Out	HealthyHostCount	Average
Disk Read, Disk Write	UnhealthyHostCount	Sample

Table 5. Experiment Set-Up

Parameters	Set-Up Case #1	Set-Up Case #2
Region	US-EAST(a)	US-EAST(b)
Instance Type	Small (Linux)	Micro (Window)
Load Balanced	Not Load Balanced	Load Balanced
AutoScaling Trigger	CPU Utilization	Latency
AutoScaling Period	30-100 Sec.	30-120 Sec.
Load Generated From	JMeter on Desktop and VMs	JMeter on EC2 Instances

4.2. Set-Up for the Case Study

The EC2 instances can be set up at different scales, including Micro, Small, Large, and Extra-Large. Besides, EC2 also supports other types of instances, such as High-CPU and High-Memory instances. Table 5 shows our detailed set-up for some EC2 instances in our case study.

Figure 4 displays the cloud infrastructure that supports the system performance and scalability evaluation for the deployed EC2 application instance. Moreover, the detailed procedure and its steps are presented here to enable cloud users to collect, monitor, and evaluate the usage of computing resources and performance parameters for a selected EC2 application instance.

- Set-up and configure an EC2 application instance through Amazon EC2 instance management API.
- Configure, set-up, and monitor system performance parameters and resource utilization metrics through *CloudWatch APIs*.

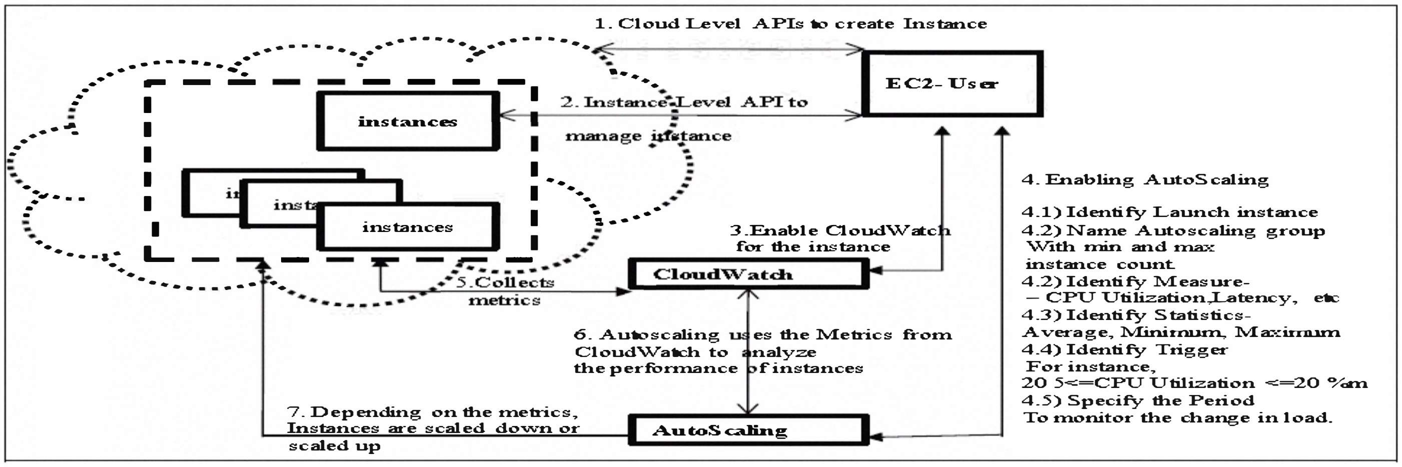

1. Cloud Level APIs to create Instance
EC2- User
instances
2. Instance Level API to manage instance
instances
3.Enable CloudWatch for the instance
4. Enabling AutoScaling
4.1) Identify Launch instance
4.2) Name Autoscaling group With min and max instance count
4.2) Identify Measure- – CPU Utilization,Latency, etc
4.3) Identify Statistics- Average, Minimum, Maximum
4.4) Identify Trigger For instance, 20 5<=CPU Utilization <=20 %m
4.5) Specify the Period To monitor the change in load.
CloudWatch
5.Collects metrics
6. Autoscaling uses the Metrics from CloudWatch to analyze the performance of instances
7. Depending on the metrics, Instances are scaled down or scaled up
AutoScaling

Figure 4. The EC2 Cloud Infrastructure and Procedure Steps for EC2 Instance Performance and Scalability Measurement

- Generate different system loads to measure the system performance and scalability for the selected EC2 application instance. VM technology and the JMeter test tool are used at client machines to simulate concurrent web user accesses and generate diverse online traffic loads.
- Both *AutoScaling API* and *CloudWatc*h facility are used to configure and monitor EC2 instance of computing resource utilization and performance data.

In this case study, we used a Java-based program (medium-size) as a sample and deployed onto an EC instance. This application program calculates the Fast Fourier Transform (FFT) for randomly selected large complex numbers. This application has a higher CPU utilization and involved numerous I/O operations. It is set-up and launched onto an EC2 instance in Amazon's cloud. Since the current EC2 cloud technology does not provide any *CloudWatch* API to monitor the allocation and utilization of memory for EC2 instances, we added and used a pre-defined Java-based memory utilization API to collect the utilization data from the launched EC2 instances.

4.3 The Case Study Results

The rest of this section reports our case study results about system performance and scalability evaluation of a selected application in Amazon EC2 instances for five perspectives: a) system resource utilization, b) system performance evaluation, c) system load measurement, d) system capacity and scalability analysis.

(a) System resource utilization

Our case study targeted at five types of computing resource utilization of the application EC2 instance using the proposed *CRUM* model as the overall utilization meter. These computing resources include EC2 CPU, disk storage, memory, network-out, and I/O operations. During the evaluation time, we applied five different traffic loads. Figure 5 shows five different polygons in the proposed CRUM model as a system resource utilization meter. Each of them presents a snapshot of the utilization of computing resources for a specific system load. The outmost polygon in blue represents shows the average utilization of the largest system load. The second outmost polygon in purple shows the average system resource usage when the second large system load is applied. As shown in Figure 5, these two polygons overlap at three extreme boundary points. This implies that both loads cause the same readings in CPU, memory, and storage. Although the outmost polygon has the largest number of system user I/O requests 175, the system network-out reading (225) is lower than the corresponding reading (241) of the second outmost polygon. The cause of this is due to the limitation of the available system resources of the targeted EC2 instance during system evaluation.

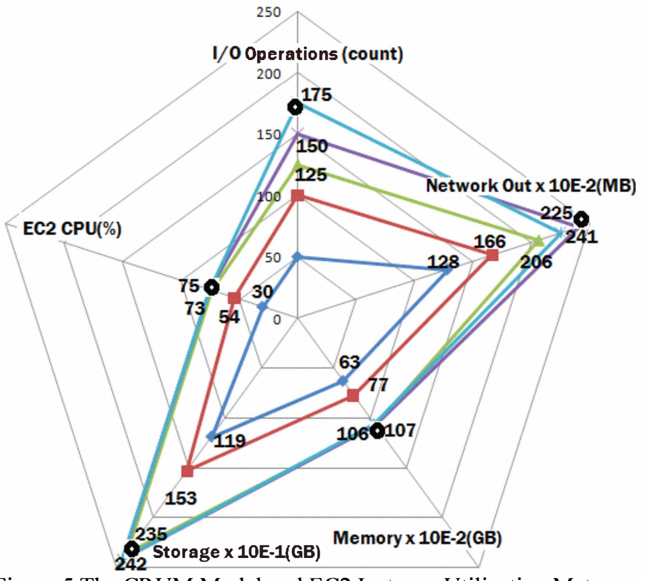


Figure 5 The CRUM Model and EC2 Instance Utilization Meter

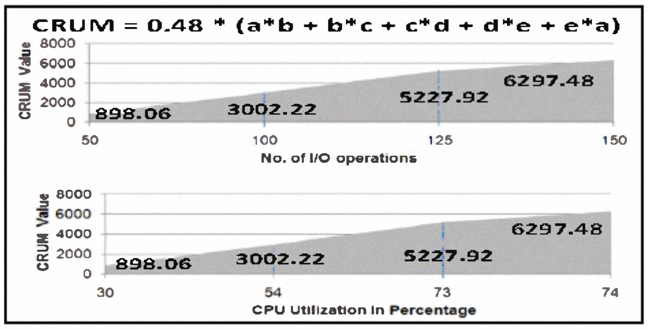


Figure 6 Two Key Performance Indicators in the CRUM model

Figure 6 supports this finding too. For example, Figure 6 shows the clear increase of the resource utilization in CPU and system-and-user I/O operations as the increase of the system load until the last two system loads.

b) System performance evaluation

To evaluate the performance of our EC2 application instance(s), we focus on five performance indicators: a) CPU utilization (%), b) availability (%), c) reliability (%), d) utilization of instance scalability (%), and e) system response time. Current Amazon's EC2 technology only supports scalability based on the pre-configured scalability conditions that which allow a user to pre-configure the number of scale-out EC2 instances from 1 to 10. Figure 7 shows the system performance meter based on the proposed performance model and metrics.

In our case study, we set up the auto scaling trigger of the CPU utilization for the EC2 instance. Its low bound is equal to 54% and its high bound is 73%. Figure 7 shows five polygons, and each of them depicts the performance of the EC2 instance under a specific system load based on five different performance indicators, including CPU utilization, system response time, reliability, availability, and the utilization of instance scalability.

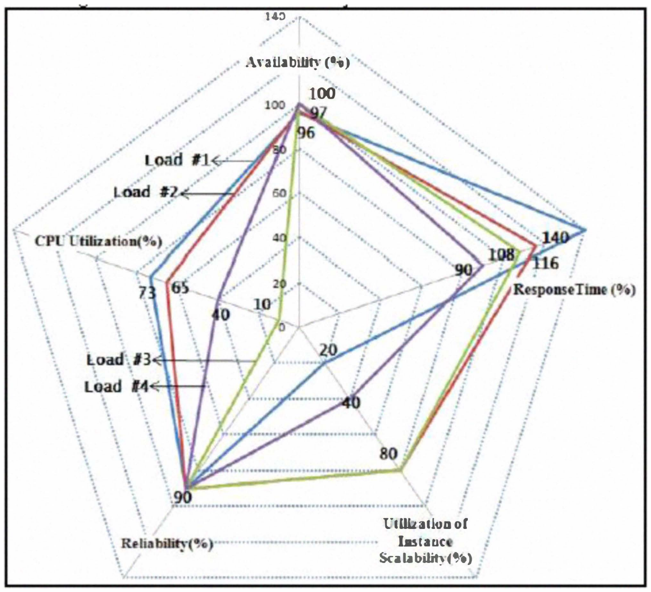


Figure 7 System Performance Meter (SPM) of the EC2 Instance

Table 6 *Load Configuration and AutoScaling*

Load Configuration	CPU Utilization	Scalability Status	No. of EC2 Related Instances
Load #1	73%	Scale-Out	2
Load #2	65%	After Scale-Out	2
Load #3	10%	Scale-Down	1
Load #4	40%	After Scale-Down	1

c) System load measurement

As shown in Table 6. We have configured and measured the EC instances with 4 different loads. Firstly, Load #1 is applied, and it causes the highest CPU utilization (73%) and a scale-out operation by adding one more EC2 instance. Next, a larger load (Load #2) is applied. Due to the scale-out factor, its CPU utilization is below the upper bound. Later, we applied a very small load (Load #3), which has only 10% of the CPU utilization. Since this is below its lower threshold (54%) of the EC2 instance, it causes the scale-down reaction of the EC2 infrastructure. Hence, the number of EC2 instances for this application is reduced back by 1. Finally, a smallest load (Load #4) is tested after the scale-down. Its corresponding CPU utilization is higher than Load #3 because a fewer number of EC2 instances are executed now. Figure 8 shows the System Load Meter using the proposed SLM model during the evaluation time based on three system load indicators, including Network-Data-In, Number of Concurrent Client Users, and Network-Date-Out. As shown in Figure 8, the blue polygon presents the maximum system load meter, and red polygon depicts the minimum system load meter. Their values are 156.28 and 6.78 respectively.

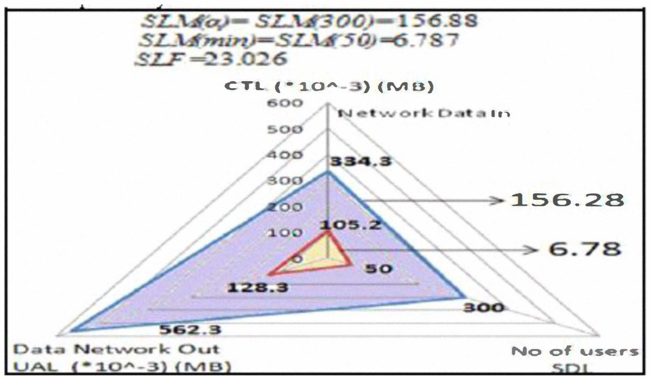


Figure 8 System Load Meter (SLM) and System Load Factor

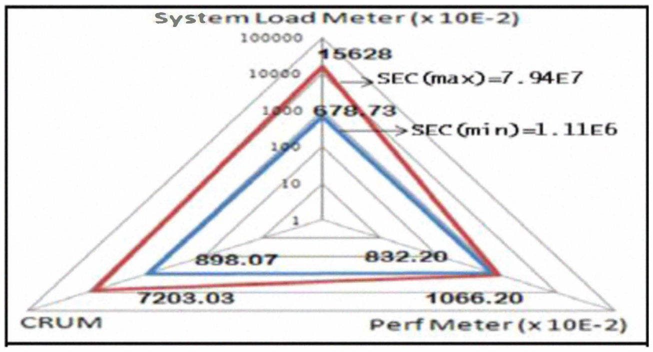


Figure 9 System Effective Capacity (SEC) Meter

d) System capacity and scalability analysis

During the case study, we also evaluate the *System Effective Capacity (SEC)* for the targeted EC2 instance using the proposed SEC model as its effective capacity meter. Figure 9 shows two different system effective capacities for the underlying EC2 instance. For instance, the red polygon shows the maximum system effective capacity (7.94E7), and the blue polygon shows the minimum system effective capacity (1.11E6).

Based on the formula in (8), the *effective scalable range (ESR)* for the targeted EC2 instance(s) can be computed below.

Effective Scalable Range (ESR):
ESR = SEC(max) –SEC(min) = (7.94E7-1.11E6) =7.8E6

4.3 Experience, Lessons, and Observations

Based on the case study, there are limitations in existing cloud technologies to support system performance and scalability evaluation. Here we summarize our experience, lessons, and observations.

- ***Need easy and effective cloud APIs and tools*** – It is important and necessary for cloud vendors to provide easy and effective cloud APIs and tools to enable cloud users (such as SaaS application vendors) to monitor, evaluate, validate application performance and scalability systematically.
- ***Watch for cloud limitations -*** There are some limitations in the APIs and tools of today's cloud technologies to support system performance and scalability evaluation and monitor. For instance, in EC2, we can't use current CloudWatch and AutoScaling facility to collect and monitor certain system resource utilization parameters from EC2 instances. The usage of memory and cache of an EC2 instance are the examples.
- ***Be aware the hidden costs in the use of cloud APIs and tools*** –Although some cloud technologies (like EC2) provide users with specific APIs and tools (for example, ***CloudWatch and AutoScaling),*** the execution of these cloud APIs and tools requires certain cloud resources and is not free of charge. For example, even though AutoScaling is free for use, it requires ***CloudWatch*** to be enabled. For EC2 instances, ***CloudWatch*** is not free for use. Hence, evaluating and monitoring the system performance and scalability for EC2 application instances is not really free.
- ***Pay attention to the inconsistent performance and scalability data –*** It is possible that we may find certain inconsistency in the collected system performance and scalability data. This has something to do with the dependency of the underlying cloud APIs and tools because their usages lead to certain system performance overheads and extra costs. For example, we found a case in which the CPU utilization reading from CloudWatch is different from the actual data collected out of the individual instances directly while the selected application is executed on a Linux flavored instance.
- ***Lack of scaling notifications for applications in clouds –*** The current cloud technology (such as EC2) does not provide any notification function and utility to support cloud users whenever any application instances (such as EC2 instances) are scaled-out, scaled-up, and scaled-down to respond the increase (or decrease) of application system loads based on

certain pre-defined scaling conditions (or thread-holds points).

- ***Third-party monitor*** - Third-party evaluation and monitor services are needed to provide reliable and accountable monitor services for SaaS application vendors and cloud vendors due to the utilization costs of computing system resources based on posted pricing models and service level agreements.

e) Related Work

In the past two decades, there are numerous published papers discussing performance evaluation and scalability analysis in conventional distributed and parallel application systems. These papers focus on four different areas: a) basic concepts, problems, and challenges [4][6][7], b) performance and scalability evaluation models [1][5][10][11][12], c) evaluation metrics and algorithms [2][3][5], and d) tools and frameworks [8][9][10].

Many of them address system performance and scalability evaluation for clusters and grid computing environments. For example, Luis Pastor et al. in [10] presented a technique for analyzing system scalability in heterogeneous systems. In [11], Ziming Zheng, et al. present a set of reliability-aware scalability models by extending Amdahl's law and Gustafson's law to consider the impact of failures on application performance.

As the fast advance of cloud computing research and technologies, many researchers and practitioners begin to pay an attention to system performance evaluation and scalability analysis for cloud-based applications. The rest of this section reviews the related research work in evaluating cloud-based application performance and scalability.

Shufen Zhang et al. in [15] highlight the evolution process of cloud computing, technologies adopted in cloud computing, as well as cloud based systems in enterprises. Meanwhile, the authors also draw a comparative study of technologies, including Amazon's Elastic Compute Cloud, Google's cloud platform, and IBM's Blue Cloud. Jian Wu et al.in [21] present a scheme for improving the scalability of service-based applications in a cloud from the granularity of the constituent services in the cloud. Moreover, they introduce a notion of scalability for service-based applications in a cloud and a framework for scalability measurement.

To provide system performance and scalability evaluation in cloud environments, effective models are needed to provide cost-effective and systematic solutions for vendors and customers. Gan-Sen Zhao et al. in [24] focus on the scalability issue due to the growth of the number of users in a cloud computing service. The paper presents a simple model for evaluating the growth of the system in terms of users. It provides a simple way to compute the scale of a system at a given time. This allows cloud computing service providers to predict the system's scale based on the number of users and to plan for the infrastructure deployment.

Since cloud-based system performance evaluation and scalability analysis requires powerful simulation solutions and environments to support large-sale data simulation and generation. This drives some research interest in VM-based simulation. M. Hasan Jamal et al. [22] discuss and analyze virtual machine (VM) scalability on multi-core systems in terms of CPU, memory, and network I/O-intensive workloads. The paper demonstrates that VMs on the state-of-the-art multi-core processors based systems scale well as multiple threads on native SMP kernel for CPU and memory intensive workloads. R. Buyya et al. [19] discuss an extensible simulation toolkit (known as CloudSim) that enables modeling and simulation of Cloud computing environments. In addition, CloudSim allows the simulation of multiple Data Centers to enable a study on federation and associated policies for migration of VMs for reliability and automatic scaling of applications.

In addition, there are a few papers discussing solutions and tools to support the evaluation of application performance and scalability in cloud environments. For example, Yong Chen et al. in [17] propose their scalability testing and analysis system (called STAS) and present its implementation with isospeed-e scalability metric. STAS provides the facility to conduct automated isospeed-e scalability measure and analysis. It reduces the burden for users to evaluate the performance of algorithms and systems. N Yigitbasi et al. in [18] report a framework, known as C-Meter, which is used to generate and submit test workloads to computing clouds. They presented their experiments using EC2 cloud technology and analyzed the performance results in terms of response time, waiting time in queue, bounded slowdown with a threshold of 1 second and job execution time.

Wickremasinghe, B. et al. in [20] present CloudAnalyst, which is a tool to simulate large-scale Cloud applications with the purpose of studying the behavior of such applications under various deployment configurations. CloudAnalyst helps developers with insights about how to distribute applications among Cloud infrastructures, the value-added services (such as the optimization of application performance), and the providers using its Service Brokers.

Unlike the existing work, this paper focuses on SaaS application performance evaluation and scalability analysis in clouds. This paper addresses this topic using uniform graphic models and formal metrics to meet the need of evaluating cloud-based SaaS performance and elastic scalability based on EC2 instances. Unlike other research work, the proposed models and metrics can be used by SaaS and/or application vendors in clouds to evaluate system performance and scalability based on system resource utilization, system performance, and system capacity. The major advantage of the proposed approach is its simplicity, uniformity, comprehensive, easy presentation and visualization. Comparing with other work, this proposes research results not only provide a simple and uniform approach to address diverse system performance and utilization indicators, but also offer a comprehensive way to analyze system capacity, scalability and its scalable range. In addition, the proposed models and metrics can be easily implemented as a part of cloud technologies or SaaS vendors to address the needs in performance evaluation, scalability testing, and monitoring. Our presented approach and case study results show its strong potential in performance evaluation and testing for cloud-based applications at the SaaS application level.

f) Conclusions and Future Work

Although there are numerous papers addressing system scalability analysis and performance evaluation, most of them focus on conventional software systems, such as distributed and parallel systems. This paper first presents the performance evaluation and scalability analysis issues and needs in clouds. Then, it introduces a set of formal and graphic models with metrics, which can be used to evaluate SaaS performance and scalability in clouds. Moreover, it reports our case study in the EC2 cloud environment. The results show the good potential application and effectiveness of the proposed models in evaluating SaaS performance and scalability.

The future extension of this research includes two folds. The first is to extend the current work to measure system performance and scalability for cloud-based applications crossing different clouds such as hybrid clouds. The next is to develop new cost-based scalability measurement models and metrics for SaaS and cloud-based applications to address the need of cost-based measurement and monitoring for cloud-based applications to confirm the given SLA and pricing models. In addition, we are developing a cloud-based performance testing and scalability analysis tool for SaaS applications based on proposed models and metrics.

Acknowledgement

This research project is supported by Tsinghua National Laboratory for Information Science and Technology (TNLIST) in 2010. It is also supported by National Science Foundation China (No. 61073003), National Basic Research Program of China (No. 2011CB302505). It is also support by a Research Grant from Fujitsu Laboratory LTD. on SaaS Application Testing on Clouds.

References

References

[1] A. Vijay Srinivas and D. Janakiram , "A Model for Characterizing the Scalability of Distributed Systems", ACM SIGOPS Operating Systems Review, Vol. 39, Issues 3, 2005.

[2] P. Jogalekar and M. Woodside, "Evaluating the scalability of distributed systems", IEEE Trans. Parallel and Distributed Systems,Vol. 6, Issue 6, pp. 589–603, 2000.

[3] Zao-Bin Gan, et al., "Evaluating the Performance and Scalability of Web Application systems Information Technology and Applications", Proceedings of 3rd International Conference on Information Technology and Applications (ICITA 2005).

[4] Lakshmi Iyer, et al., "Performance, scalability and reliability issues in web applications", Industrial Management and Data Systems, Vol. 105, Issue 5, pp. 561-576, 2005.

[5] Xian-He Sun, et al., "Scalability of Heterogeneous Computing", Proceedings of International Conference on Parallel Processing (ICPP 2005).

[6] X ingfu Wu, Performance Evaluation, Prediction and Visualization of Parallel Systems, Springer; 1999.

[7] Vipin Kumar, Anshul Gupta, "Analyzing the scalability of parallel algorithms and architectures: A survey". Proceedings of the 1991 International Conference on Supercomputing, June 1991.

[8] G. Lyon, R. Kacker, and A. Linz, "A scalability test for parallel code", Software: Practice and Experience, Vol. 25, Issue 12, 1995.

[9] Leticia Duboc, David S. Rosenblum, and Tony Wicks,"A Framework for Modeling and Analysis of Software Systems Scalability", ICSE'06, 2006, Shanghai, China.

[10] Luis Pastor, et al,"An Efficiency and Scalability Model for Heterogeneous Clusters", Proceedings of the 3rd IEEE conference on Cluster Computing (2001).

[11] Ziming Zheng and Zhiling Lan,"Reliability-Aware Scalability Models for High Performance Computing", Proceedings of IEEE International Conference on Cluster Computing and Workshops (CLUSTER '09).

[12] J. Perez, C. Germain-Renaud, B. Kégl, "Responsive elastic computing", Proceedings of the 6th International Conference Industry Session on Grids meets Autonomic Computing (2009)

[13] T Dillon, et al., "Cloud Computing: Issues and Challenges", Proceedings of 24th IEEE Conference on Advanced Information Networking and Applications (AINA 2010).

[14] Lijun Mei, et al., "A Tale of Clouds: Paradigm Comparisons and Some Thoughts on Research Issues", Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference (APSCC 2008),

[15] Shufen Zhang, et al., "Analysis and Research of Cloud Computing System Instance", Proceedings of 2nd International Conference on Future Networks (ICFN '10).

[16] Sean K. Barker and Prashant Shenoy, "Empirical Evaluation of Latency-sensitive Application Performance in Cloud", Proceedings of the first annual ACM SIGMM conference on Multimedia systems, 2010.

[17] Y. Chen and X. Sun, "STAS: A Scalability Testing and Analysis System", Proceedings of IEEE International Conference on Cluster Computing, 2006.

[18] N Yigitbasi, et al., "C-Meter: A framework for Performance Analysis for Computing clouds ", Proceedings of 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '09).

[19] Buyya, R, et al., "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities", Proceedings of International Conference on the High Performance Computing & Simulation (HPCS 09).

[20] Wickremasinghe, B. et. al, "CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications", 24th IEEE International Conference on Advanced Information Networking and Applications (AINA), 2010.

[21] Jian Wu, et al., "Improving Scalability of Software Cloud for Composite Web Services" Proceedings of International Conference on Cloud Computing, CLOUD '09.

[22] M. Hasan Jamal, et al., "Virtual Machine Scalability on Multi-Core Processors Based Servers for Cloud Computing Workloads", Proceedings of IEEE International Conference on Networking, Architecture, and Storage (2009).

[23] Jerry Zeyu Gao, H.-S. Jacob Tsao, Ye Wu, Testing and Quality Assurance for Component-Based Software, Artech House Publishers, 2003.

[24] Gan-Sen Zhao, et al., "Modeling User Growth for Cloud Scalability and Availability", International Journal of Technology-Chinese Vol. 1 No. 1 (2010/5).

[25] Chambers, John, William Cleveland, Beat Kleiner, and Paul Tukey, (1983). *Graphical Methods for Data Analysis*. Wadsworth. pp. 158-162.

[26] Jerry Gao, Xiaoying Bai, and Wei-Tek Tsai, "Cloud Testing-Issues, Challenges, Needs and Practices", Software Engineering: An International Journal, Volume 1, Number 1, September, 2011.

[27] L. Riungu, O. Taipale, and K. Smolander, "Research Issues for Software Testing in the Cloud", proceedings of 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), 2010, pp. 557–564.

[28] L. Riungu, O. Taipale, and K. Smolander, "Software Testing as an Online Service: Observations from Practice," Proceedings of Third International Conference on Software Testing, Verification, and Validation Workshops, 2010, pp. 418–423.

[29] G. Candea, S. Bucur, and C. Zamfir, "Automated Software Testing as a Service", Proceedings of the 1st ACM symposium on Cloud Computing, 2010, pp. 155–160.