Cloud Performance Modeling with Benchmark Evaluation of Elastic Scaling Strategies

Kai Hwang, *Life Fellow, IEEE*, Xiaoying Bai, *Member, IEEE*, Yue Shi, Muyang Li, Wen-Guang Chen, and Yongwei Wu, *Member, IEEE*

Abstract—In this paper, we present generic cloud performance models for evaluating Iaas, PaaS, SaaS, and mashup or hybrid clouds. We test clouds with real-life benchmark programs and propose some new performance metrics. Our benchmark experiments are conducted mainly on IaaS cloud platforms over scale-out and scale-up workloads. Cloud benchmarking results are analyzed with the efficiency, elasticity, QoS, productivity, and scalability of cloud performance. Five cloud benchmarks were tested on Amazon IaaS EC2 cloud: namely YCSB, CloudSuite, HiBench, BenchClouds, and TPC-W. To satisfy production services, the choice of scale-up or scale-out solutions should be made primarily by the workload patterns and resources utilization rates required. Scaling-out machine instances have much lower overhead than those experienced in scale-up experiments. However, scaling up is found more cost-effective in sustaining heavier workload. The cloud productivity is greatly attributed to system elasticity, efficiency, QoS and scalability. We find that auto-scaling is easy to implement but tends to over provision the resources. Lower resource utilization rate may result from auto-scaling, compared with using scale-out or scale-up strategies. We also demonstrate that the proposed cloud performance models are applicable to evaluate PaaS, SaaS and hybrid clouds as well.

Index Terms—Cloud computing, performance evaluation, cloud benchmarks, and resources scaling

1 Introduction and Motivation

The hype of cloud computing is entering the disillusionment stage, reaching the plateau of productivity in the next decade. Following a pay-as-you-go business model, cloud platforms are gradually adopted by the main stream of IT industry. Cloud computing attempts to provide an integrated platform to benefit many users at the same time. This multi-tenant and on-demand service model is achieved through virtualization on all shared utilities and resources [6], [21].

This paper models cloud performance for IaaS, PaaS and SaaS clouds at different abstraction levels. We assesse various benchmarks targeted at clouds, and analyze new performance results. We assess the state of cloud computing from the perspectives of performance. This work is extended from previous works by [2], [3], [4], [7], [8], [9], [10], [11], [15], [16], [20], [22], [23], [24], [25], [26], [27], [30], [31], [32], [33], [34], [35], [36], [37].

Up to now, the original cloud design goals are only partially fulfilled. We are still climbing a steep hill to deliver sustained cloud productivity. To reduce the cost of leased resources and to maximize utilization, elastic and dynamic resource provisioning are the foundation of cloud performance.

NIST [28] has identified that cloud computing demands scalable performance, economics of scale, measurable productivity, high availability and energy efficiency. With guaranteed SLA (*service-level agreement*), cloud automatically allocates more resources by *scale-up or scale-out* resources [14], [29], when the workload increases beyond certain threshold. The system releases unused resources by *scale-down* or *scale-in* [5], [19], [30] when the load reduces.

Cloud scaling is enabled with virtualized resources. Hence, the scale of computing power is calculated at the abstraction level of virtual resources. To handle workload composed of large number of small jobs, performance concerns are the average response time and throughput, rather than completion time of individual tasks. Hence, scalability needs to upgrade the system capability to handle large number of small users. Cloud productivity is tied to the performance cost ratio.

To meet the demand, we propose some new cloud performance metrics in terms of efficiency and productivity. The resources are scaled by the quantity and types of virtual machine instances. The scalability is driven by cloud productivity, taking both QoS and price into consideration. Various benchmark suites have been suggested to evaluate cloud performance under different workloads in the past.

We chose a set of widely used cloud benchmarks to test the scale-out and scale-up capabilities of EC2-like cloud platforms. The workload patterns include large scale data processing and data analytics, web search and service. Five benchmarks applied on EC2 include the BenchCloud at USC, YCSB from CloudSuite [14], HI Bench [20], and TPC-W [35].

- *K. Hwang is with the Departments of Electrical Engineering and Computer Science, University of Southern California, Los Angeles, CA, 90089. E-mail: kaihwang@usc.edu.*
- *X. Y. Bai is with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. E-mail: baixy@ tsinghua.edu.cn.*
- *Y. Shi is with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA, 90089. E-mail: yueshi@usc.edu.*
- *M. Y. Li, W. G. Chen, and Y. W. Wu are with Department of Computer Science and Technology, Tsinghua University, Beijing, China. E-mail: limuyang08@gmail.com, {cwg, wuyw}@ tsinghua.edu.cn.*

Manuscript received 31 July 2014; revised 17 Dec. 2014; accepted 14 Jan. 2015. Date of publication 29 Jan. 2015; date of current version 16 Dec. 2015.
Recommended for acceptance by D. Padua.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2015.2398438

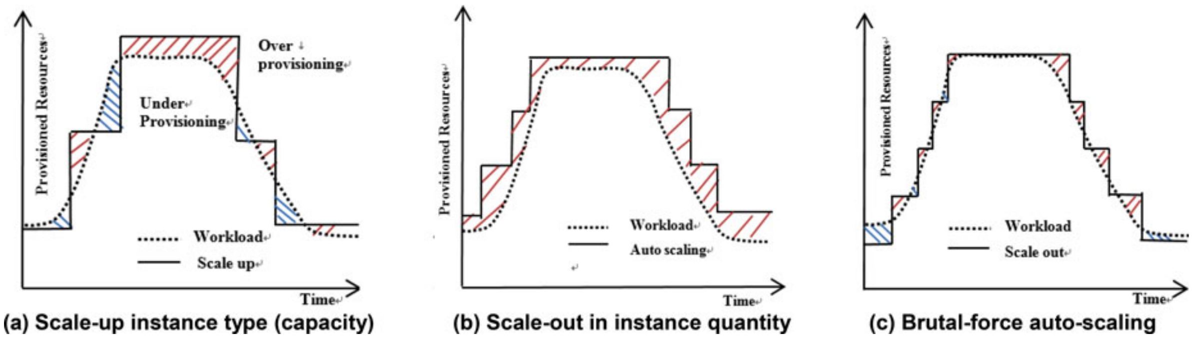


Fig. 1. Auto-scaling, scale-out and scale-up machine instance resources in elastic IaaS clouds, where over-provisioning and under-provisioning of machine resources are shown in differently shaded areas above and below the workload curves.

Cloud relies on virtualization technique to enable elastic resource provisioning or de-provisioning. Hence, the effectiveness of virtualization becomes crucial to cloud performance. From the software perspective, multi-tenant architecture is introduced with clouds to support big-data processing and Internet/web services.

In the remaining sections, we cover cloud workload, benchmarks tested and performance metrics including some newly discovered ones. Then we provide an elasticity analysis and study the interplay between efficiency, productivity and scalability of cloud platforms. We also reveal the tradeoffs between scaling out and scaling up policies. Our benchmark experiments were conducted on Amazon EC2.

2 Scaling Strategies and Benchmark Suites

Due to multi-tenant demands, clouds are facing all sorts of workloads including multi-tasking, batch processing, streaming, data-mining and analytics. The cloud workload must be matched with adequately configured resources to achieve high performance and sustained productivity.

2.1 Auto-Scaling, Scale-Out, Scale-Up and Mixed Strategies

Clouds are used primarily for data-intensive and latency-sensitive jobs, search engines, OLTP/business processing, social-media networking, data warehousing and big-data analytics. Cloud workloads are characterized by their dataset size, algorithms, memory-access pattern, and service model applied. We demonstrate three cloud resource scaling techniques in Fig. 1.

The *Auto scaling* shown in Fig. 1c is a brutal-force strategy to increase or decrease resources in a cloud. The idea is to add more machine instances when a specific resource (like CPU) utilization rate exceeds a preset threshold during a fixed observation period. Practicing auto scaling can enhance the cloud performance at the expenses of always provisioning more resources above the workload demand.

As seen from Fig. 1, auto-scaling is easy to implement with a utilization threshold approach. However, it tends to waste higher in over-provisioned resources. We illustrate the ideas of scaling up resources in Fig. 1a and scaling-out resources in Fig. 1b. These scaling strategies and their possible mixtures are characterized below:

- ***Auto-scaling*** strategy applies a threshold to increase the machine instance automatically, once the instance utilization rate exceeds a preset threshold (say 85 percent) for a preset period (say 60 sec). Auto-scaling tends to over-provision resources to satisfy the user at run time.
- ***Scale-out*** strategy allows adding more machine instances or processing nodes of the same type based on the quota agreed in the SLA. Obviously, scaling out appeals more to the use of homogeneous clusters with identical nodes.
- ***Scale-up*** strategy is implemented with scaling the cloud from using small nodes to more powerful nodes with better processor, memory or storage.
- ***Mixed scaling strategy*** allows one to scale up (or scale-down) the instance type and adjust the instance quantity by scale-out (or scale-in) resources at the same time. Mixed scaling appeals better with using heterogeneous clusters.

We will evaluate the relative performance of the three scaling strategies in subsequent sections. In general, the scale-up approach (Fig. 1a) takes longer overhead to reconfigure and has the lowest elasticity among all scaling approaches. Scaling-up or down take longer time and thus results in both over-provisioning and under-provisioning of resources as seen by the shaded areas above or below the workload curve.

Scale-out strategy (Fig. 1b) matches the workload variation closely. Thus it has the lowest over- or under-provisioning of resources. The auto-scaling (Fig. 1c) wastes resources in over-provisioning. But it will not cause any interruption in client services committed. This is the main reason why scale-out is more often practiced in cloud platforms than the scale-up approach.

2.2 Cloud Benchmark Suites Tested

Table 1 summarizes five open-source cloud benchmark suites we have tested. The Yahoo YCSB [11] and TPC-W [34] are developed by industry. The BenchClouds and CloudSuite are developed in the academia. The CloudSuite [14] was developed at EPFL in Lausanne. All source codes and datasets are available in these open-source benchmarks. The BenchCloud is still under development at USC. This suite collects users programs and datasets mainly from social-media and big data applications.

HiBench [20] is specifically tailored for testing Hadoop application on most clouds. The suite was developed for measuring the speed, throughput, HDFS bandwidth, and resources utilization in a large suite of programs. The YCSB is a *Yahoo! Cloud Serving Benchmark*. Other cloud benchmarks

TABLE 1
Cloud Benchmarks, Workloads, Metrics Applied, and Systems Tested

Benchmark and Reference	Reported Applications and Workloads	Performance Metrics	Clouds Applied and Workload Generation
BenchCloud under development at USC	Social-media applications with big-data processing	Speedup, Efficiency, QoS, Scalability	AWS EC2, Twitter API-workload
CloudSuite at EPFL, Lausanne [14]	Data/Graphics analytics, Media Streaming and Web Services	Latency, WIPS, Speedup, Efficiency, Scalability	AWS, GAE, Faban workload generator
Hi-Bench at Intel [20]	Terasort, Word count, DFSIO, Nutch indexing, Page Rank, etc.	Speed, HDFS bandwidth, utilizations (CPU, memory, IO)	Hadoop Random TextWriter, TeraGen, enKMeansDataset
TPC-W by Transaction Proc.Council [34]	Web Search, and Analytical Query Processing	WIPS, $/WIPS, TPS, QoS, Efficiency	AWS EC2, Rackspace, TPC client workload
YCSB by Yahoo! [11]	Synthetic workload, data services,	Latency, Throughput, Speedup, Scalability, Replication Impact	Microsoft Azure, AWS, HBase, Shared MySQL

include the CloudCmp [27], Phoronix [17], CloudStone [35], and C-meter [37], which were not tested in our experiments.

Interested readers are refer to the assessment by Farber and Kounev [13] for cloud benchmarking trends. Two commercial cloud evaluations were conducted recently. Nine public clouds were evaluated by BitCurrent [4] and 144 cloud sites were examined by CloudHarmonics [10]. However, the performance metrics they have applied are far from being adequate to cover the QoS and productivity in clouds. Performance in cloud environment.

3 Cloud Performance Metrics

We apply an extended concept of performance to include capabilities and productivity. The performance and capabilities are necessary to upgrade the productivity of a cloud. In Table 2, we divide cloud performance metrics at three levels: namely *performance, capabilities* and *productivity*.

3.1 Three Performance Levels

Basic performance metrics include most traditional metrics such as speed, speedup, efficiency, utilization, etc. [11], [21], [22], [23]. Cloud capabilities are marked by network latency, data throughput, storage capacity, data analytics, and system recoverability. The third level deals with cloud productivity, which is revealed by QoS, SLA, security, Power, cost and availability, etc. Table 2 summarizes these metrics in 3 groups.

Most basic performance metrics and capability measures have defined in the past. Some elasticity, productivity and scalability measures are newly proposed here. We will demonstrate the power of using those new metrics in evaluating cloud performance in subsequent sections.

3.2 Basic Performance Metrics

These include traditional performance measures of speed, speedup, efficiency, etc. for parallel and distributed computing. More in-depth definitions can be found in [12], [21], [22], [23].

- *Speed* (S)*:* Number of *millions of operations per second* (*Mops*). The operation could be integer or floating-point like *MFlops.* The speed is also known as *throughput* by TPC-W benchmark, measured by *millions of web interactions per second* (*WIPS*).

TABLE 2
Performance, Capability, and Productivity Metrics for Evaluating Clouds

Abstraction Level	Performance Metric	Notation (Eq. #)	Brief Definitions with Representative Units or Probabilities
Basic Performance Metrics	***Execution time***	T_e	Time elapsed during program or job execution, (sec., hours)
	Speed	S_r	Number of operations executed per second, (PFlops, TPS, WIPS, etc.)
	Speedup	S_u	Speed gain of using more processing nodes over a single node
	Efficiency	E	Percentage of max. Performance (speedup or utilization) achievable (%)
	Scalability	S (Eq. (5))	The ability to scale up resources for gain in system performance
	Elasticity	E_l, (Eq. (14))	Dynamic interval of auto-scaling resources with workload variation
Cloud Capabilities:	***Latency***	T	Waiting time from job submission to receiving the first response. (Sec.)
	Throughput	H	Average number of jobs/tasks/operations per unit time (PFops, WIPS.)
	Bandwidth	B	Data transfer rate or I/O processing speed, (MB/s, Gbps)
	Storage Capacity	S_g	Storage capacity with virtual disks to serve many user groups
	Software Tooling	S_w	Software portability and API and SDK tools for developing cloud apps.
	Bigdata Analytics	A_n	The ability to uncover hidden information and predict the future
	Recoverability	R_c	Recovery rate or the capability to recover from failure or disaster (%)
Cloud Productivity	***QoS of Cloud***	QoS	The satisfaction rate of a cloud service or benchmark testing (%)
	Power Demand	W	Power consumption of a cloud computing system (MWatt)
	Service cost	$Cost$	The price per cloud service (compute, storage, etc.) provided, ($/hour)
	SLA/Security	L	Compliance of SLA, security, privacy or copyright regulations
	Availability	A	Percentage of time the system is up to deliver useful work. (%)
	Productivity	P, (Eq. (4))	Cloud service performance per unit cost, (TFlops/$, WIPS/$, etc.)

- *Speedup* (S_u): Speed gain of using multiple nodes
- *Efficiency* (E_f): Percentage of peak performance achieved
- *Utilization* (*U*): Busy resources (CPU, memory, storage).
- *Scalability* (S_c)*:* Scaling ability to upgrade performance.

3.3 Cloud Capabilities and Productivity

These are macroscopic metrics that describe the hardware, software, reconfiguration and networking capabilities of a cloud as listed below: These metrics are good indicators of cloud's performance basis.

- *Latency* (*L*): System response time or access latency
- *Bandwidth* (*B*): This is data transfer rate or I/O rate.
- *Elasticity* (E_l): The ability for cloud resources to scale up/down or scale in/out to match with workload variation
- *Software* (S_w): Software portability, API and SDK tooling
- *Big-data Analytics:*(A_n)*:* The ability to uncover hidden information or predict trends in big data.

For the first time, we introduce cloud productivity as a compound function of QoS, availability, power efficiency, and performance-cost ratio. These attributes are defined below. More details are given in subsequent sections.

- *Quality of Service* (*QoS*): Satisfaction on user services
- *System availability* (*A*)*: The* system up time per year.
- *Service costs* (C_o): User renting costs and provider cost.
- *Power Demand* (*W*)*:* Cloud power consumption (MWatt).
- *SLA/Security* (*L*)*:* Compliance of SLA, security, etc.
- *Productivity* (*P*)*:* QoS-satisfied performance per unit cost.

4 Efficiency, Productivity and Scalability

In this section, we analyze three mutually related factors toward productive cloud performance. The scalability concept was developed with the parallel computing community [22]. The elasticity was introduced with the inception of cloud computing [18]. Productivity of clouds is newly introduced in this paper extending our preliminary work reported in CloudCom 2014 [23]. We attempted to relate cloud productivity to QoS-satisfied performance over business gains in cloud computing systems.

4.1 Cloud Efficiency and Productivity

We specify a cloud configuration on the resources provisioned at a given time instance. The configuration is described by a resources matrix $\Lambda = [a_{ij}]_{m\times k}$ as follows.

$$\Lambda = \begin{array}{c} \\ r_1 \\ r_2 \\ \cdots \\ r_m \end{array} \begin{pmatrix} v_1 & v_2 & \dots & v_k \\ a_{11} & a_{12} & \dots & a_{1k} \\ a_{21} & a_{22} & \dots & a_{2k} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mk} \end{pmatrix}. \tag{1}$$

In this resource matrix, we have

1) $V = \{v_j | j = 1, 2, \dots, k\}$ are machine instances;
2) $R = \{r_i | i = 1, 2, \dots, m\}$ are resources types in instances;
3) $a_{ij} = \{1 \le i \le m, 1 \le j \le k\}$ are resource quantity.

Consider a cluster configuration Λ. Let *T*(1) be the execution time of an application code on a 1-ECU instance. Let $T(\Lambda)$ be the execution time of the same code on a virtual cluster Λ. The speedup is defined by *Speedup* $(\Lambda) = T(1)/T(\Lambda)$. Assume that the cluster is built with *n* instance types. The type-*I* has n_i instances, each with an ECU count c_i. We calculate the total cluster ECU count by:

$$N(\Lambda) = \sum_{i=1}^{i=n} n_i \times c_i. \tag{2}$$

This $N(\Lambda)$ count sets a ceiling of the cluster speedup. Now, we are ready to define the *cloud efficiency* for the cluster Λ in question as follow:

$$\begin{aligned} Efficiency(\Lambda) &= Speedup(\Lambda)/N(\Lambda) \\ &= T(1) \Big/ \left\{ T(\Lambda) \times \sum_{i=1}^{i=n} n_i \times c_i \right\}. \end{aligned} \tag{3}$$

In general, the cloud *productivity* is driven by three technical factors that are related to the scaling factor.

1) System performance such as throughput in terms of transactions per second or response time.
2) System availability as an indicator of QoS measured by percentage of uptime.
3) Cost for rented resources measured by price.

Let Λ be a cloud configuration in use. We define the cloud *productivity* by three factors, all are functions of Λ.

$$P(\Lambda) = \frac{p(\Lambda) \times \omega(\Lambda)}{C(\Lambda)}, \tag{4}$$

where $p(\Lambda)$ is a *performance* metric used, which could the speed or throughput selected from Table 2. The $\omega(\Lambda)$ is the *QoS* of the cloud. For simplicity, one can approximate the QoS by the *service availability* measure. According to CloudHarmony Report on 144 public clouds [10], more than half have 99 percent or higher availability. The$C(\Lambda)$ represents user cost to rent resources in forming the virtual cluster Λ.

4.2 Production-Driven Scalability

For different workload, scalable performance is often tied to different resource types, even though instances are often provisioned in configuration package. The performance of CPU-bound jobs is primarily decided by machine instance numbers. Memory-bound problems are limited by the memory (including cache) allocated within the machine instances. The storage-bound problems are limited by the network latency and disk storage and I/O bandwidth encountered.

The *cloud scalability* is driven by the productivity and QoS of a cloud system. This measure is inversely proportional to the service costs As we scale from configuration $\Lambda1$ to another $\Lambda2$. This metric evaluates the economy of scale by a pair of productivity ratio. The higher is the value of a scalability measure, the more opportunity exists to target the

TABLE 3
Machine Instance Types in Amazon EC2 in 2014

Instant Type	ECU	Virtual Cores	Memory (GB)	Storage (GB)	Price ($/hour)
m1.small	1	1	1.7	1 × 160	0.044
m1.medium	2	1	3.7	1 × 410	0.087
m3.medium	3	1	3.75	1 × 4 SSD	0.07
m1.xlarge	8	4	15	4 × 420	0.350
m3.xlarge	13	4	15	2 × 40 (SSD)	0.280
c1.xlarge	20	8	7	4 × 420 (SSD)	0.520
c3.xlarge	14	4	7.5	2 × 40 (SSD)	0.210

desired scaling scheme.

$$S(\Lambda 1, \Lambda 2) = \frac{P(\Lambda 2)}{P(\Lambda 1)} = \frac{p(\Lambda 2) \times \omega(\Lambda 2) \times C(\Lambda 1))}{p(\Lambda 1) \times \omega(\Lambda 1) \times C(\Lambda 2)}. \quad (5)$$

With comparable QoS and cost estimation, the scalability is directly proportional to productivity (Eq. (4)). Therefore, will demonstrate the measured productivity results and skip the scalability plots in subsequent sections.

Table 3 shows some machine instances applied in our experiments on EC2. The provider rents resources by instance types and quantity. AWS has defined a term ECU (*EC2 Compute Unit*) as an abstract unit to quantify the computing power of each instance type. By 2009 standard, the performance of a 1 ECU instance is equivalent to a CPU built with 1.2 GHz 2007 Xeon processor [1]. The memory and storage capacity also affect the ECU count. For example, a system may rent three instances on EC2 for general purpose applications with two instance types. We use an instance vector $V = \{m1.large, m3.large\}$ built with $a_{m1.large} = 1$ and $a_{m3.large} = 2$ instances. To assess the cost effectiveness, we list also the instance renting prices in 2014.

5 Cloud Performance Models

Depending on the cloud service models applied, the resources could be controlled by users, vendors, or by both jointly. As a comparison, control of desktop computing systems falls in the hands of users, except the control of networking facility which is shared. This adds a great burden on the part of users. The control of cloud resources shifts the burden from users to vendors as we change to IaaS, PaaS, and SaaS clouds.

5.1 Generic Cloud Performance Model

First, we introduce a generic cloud performance model. Then we will show how to extend or refine the generic framework to model all types of cloud computing services. The performance of a cloud, denoted as ***F***(*Cloud*), is modeled by a *performance function* ***F***, consisting of a 5-tuple expression.

$$\begin{aligned} F(Cloud) = \{&Service\ Model,\ Service\ Offerings,\\ &Performance, Capabilities, Availability\}, \end{aligned} \quad (6)$$

where the *Cloud* is identified by the cloud site name. The *Service Model* could be one or more of the available service modes such as IaaS, PaaS, SaaS, DaaS (*Data as a Service*), TaaS (*Testing as a Service*), HaaS (*Health-care as a Service*), NaaS (*Network as a Service*), LaaS (*Location as a Service*), CaaS (Communication as a Service), etc.

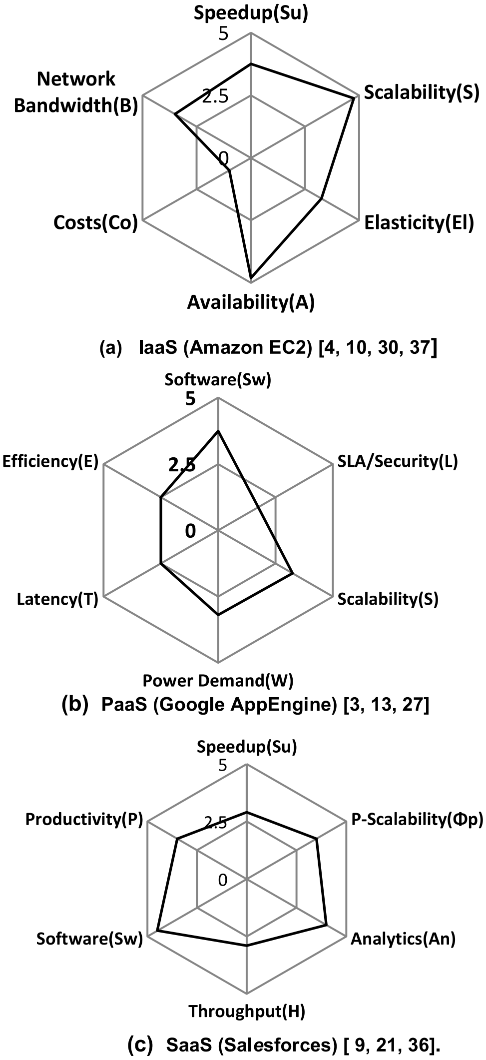


Fig. 2. Performance maps of three representative platforms for IaaS, PaaS and SaaS clouds, where the polygon data points are extracted from the cited reports on Amazon EC2, Google AppEngine, and Salesforce cloud.

The *performance* here refers to a subset of performance metrics selected from Category 1 in Table 2. To illustrate the modeling ideas, we first specify three basic cloud service models, namely IaaS, PaaS and SaaS. Then we show how to extend the model to cover hybrid clouds or cloud mashups.

5.2 IaaS Performance Model

We test the following set of performance-attributes in evaluating an IaaS cloud. This model specification could be specially tailored to special user groups or providers. Fig. 2 shows three Keviate charts for three cloud service models. Each spoke of the polygon represents an attribute dimension. The attribute scale is proportional to the directional length along the spoke. The further away from the center, the higher performance is expressed in a scale from 0 to 5. Where value "0" means the least performance and "5" the highest.

The polygon area offers an average or approximated indicator of the overall performance of the cloud along those dimensions. Let $\{p_i | i = 1, 2, \ldots, n\}$ be a set of n performance attributes. In general, the larger is the area of the polygon (Eq. (7)), the higher is the average performance demonstrated. Here we assume that all six dimensions are equally weighted.

$$Area = 0.5 \times \sin(2\pi/n) \times \sum p_i \times p_{i+1}. \tag{7}$$

Three representative cloud configurations are modeled in Fig. 2 along different sets of performance metrics. They differ in resources provisioned, performance level achieved, performance results recorded, etc. In general, we suggest the following 5-tuple to model the performance of an infrastructure IaaS cloud:

$$F(Infrastructurecloud) = \{< IaaS >, < Compute, Storage >, < S_u, E_{l,} S >, < B >, < A, C_o >\}, \tag{8}$$

where six metrics are selected from Table 2. Fig. 2a shows the Amazon EC2 performance map, where the polygon data points are extracted and normalized from previous reports in [4], [10], [30], [37]. With some modifications, the model can be applied to evaluate other IssS clouds like Rackspace, GoGrid, FlexiScale, and Joyent [21].

5.3 PaaS and SaaS Cloud Performance

The PaaS cloud platforms are used mainly in developing user applications. Therefore, Eq. (9) a special set of performance metrics are selected, different from those used to evaluate IaaS models. For application developers, the major concern is programmability or the effective use of *software development kits* (SDK), etc. as in Fig. 2b. Again the dimensional performance is based on previous reports [3], [13], [27].

$$F(Platform\ Cloud) = \{<PaaS>, <Apps\ Development, TaaS >, < E_, S >, < B, S_w>, < W, L >\}, \tag{9}$$

where the six performance metrics are selected from in Table 2. This model can modified to evaluate many PaaS platforms like Microsoft Azure, Google AppEngine, and Salesforce Force.com, Amazon Elastic MapReduce, and Aneka [21].

Multi-tenant architecture is reflected in a SaaS model. It allows for a single software instance to be shared by many tenants. Each user may work in a dedicated environment. Listed below are commonly concerned issues that relate to SaaS performance. For simplicity, we show in Eq. (10) the SaaS map model in six performance dimensions.

$$F(Application\ Cloud) = \{< SaaS >, < Marketing, Social\ Media >, < Su, \Phi_p>, < H, S_{w,} A_n>, < P >\}, \tag{10}$$

where six metrics are selected from Table 1. In Fig. 2c, we plot two performance polygons for Salesforce in CRM (*customer relation management*) applications. The data are points extrapolated from [9], [21], [36]. This model can be modified to evaluate many SaaS clouds like Google Gmail, IBM Lotus Live, Microsoft Dynamic CRM, and Salesforce CRM, etc.

5.4 Modeling Hybrid Clouds or Mashups

Private clouds are used by organization or enterprise employees. They are used for research/development or providing messaging or CaaS (*Communication as a Service*), etc. Private clouds have better security, cost factors and availability. Private cloud users are more concerned about raw speed, utilization and productivity, etc.

Hybrid clouds are built with private cloud interacting closely with some public clouds. They are also known as *cloud mashups*. Given below in Eq. (11) is an example performance model for hybrid clouds or mashups.

$$F(Hybrid\ Cloud) = \{< IaaS, PaaS, SaaS >, <SocialMedia, Compute, BackupStorage, etc.>, <S_u, U, E, \Phi, S_r, T_e>, <T, H, B, S_g, S_w>, <A, C_o>\}. \tag{11}$$

This model in Eq. (12) compares the relative performance of several benchmark suites running on the same cloud platform. This model was applied to compare the performance of Hi Bench and BenchClouds in Fig. 11a.

$$F(YCSB, CloudStone, BenchCloud) = \{<AWSEC2\ andS3>, <YCSB, CS, BC >, < Raw\ speed(S_r), Utilization(U), Service\ Costs(C_0), Productivity(P)>\}. \tag{12}$$

Consider k cloud platforms $<C_1, C_2, \ldots, C_k>$. Which are under the test by p benchmark programs $<B_1, B_2, \ldots, B_p>$. Assume that the clouds are tested by m performance metrics $<M_1, M_2, \ldots, M_m>$. The following model Eq. (13) reveals the relative performance of multiple cloud platforms. For example, EC2 and Rackspace are evaluated in Fig. 11b for the case of choosing $k = 2$, $p = 1$ and $m = 6$.

$$F(C_1, C_2, \ldots, C_k) = \{<C_1, C_2, \ldots, C_k>, <B_1, B_2, \ldots, B_p>, <M_1, M_2, \ldots, M_m>\}. \tag{13}$$

6 Elasticity of Cloud Performance

Elasticity in computer systems cannot be achieved without virtualization. Multi-tenancy cloud architecture demands elastic resources with auto-scaling to yield scalable performance. Differences in abstraction levels (IaaS, PaaS, SaaS) affect the system reconfiguration capability or the elasticity of clouds. In the past, physical computer resources may take hours or days to reconfigure. Thus, the elasticity is very low due to large reconfiguration overhead.

The elasticity was introduced by Herbst et al. [18] to evaluate cloud scalability from two perspectives : (1) How fast or timely to change the resources state in a cloud? (2) How precisely the resources are provisioned to address the workload variations? Elasticity has made possible to reconfigure within a very short time by machine virtualization.

This concept is illustrated in Fig. 3, where the elasticity is measured with two parameters: *speed* and *precision*. *Speed* is calculated by the time delay (θ) of the provisioning or de-provisioning process, while precision is the offset (μ) with under- or over-provisioning. The concept

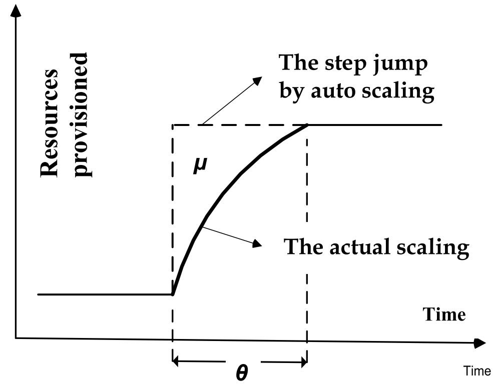


Fig. 3. Illustration of cloud resource provisioning, where θ is the overhead time and μ is the offset between actual scaling and the auto scaling process.

of elasticity is illustrated in Fig. 3 in connection with these two parameters.

Elasticity defines the degree to which a system is able to adapt to workload changes by provisioning and de-provisioning resources in an autonomic manner, such that at each time the available resources match the current demand as closely as possible". Let θ be the average time to switch from an under-provisioned state to an elevated state and μ be the offset between actual scaling and the auto scaling. The elasticity is defined by the following expression:

$$E_l = 1/(\theta \times \mu). \tag{14}$$

Fig. 4 plots the elasticity as a function of the reconfiguration overhead (θ) under different provisioning offsets (μ) from the actual scaling curve. When the offset is small ($\mu = 10\%$), the elasticity drops sharply as the overhead (θ) increases. When the offset gets to 70 percent, the elasticity drops to 0.04 from 0.25, when the average provisioning time θ is at 40 sec. Then the elasticity stay rather low flatly as θ increases.

The message being conveyed here is that in order to increase the elasticity of a cloud system, we should minimize the provisioning time and keep the provision offset as low as possible. The elasticity is a necessary condition for scalability, but not sufficient. The built-in auto-scaling mechanism (illustrated in Fig. 3) is greatly affected by the elasticity measure. The fluctuation of resource usage and the delay of instance replication or upgrading are all affecting the performance in cloud applications.

7 Measured Cloud Benchmark Results

We have performed extensive cloud benchmark experiments on AWS EC2 with EMR (Elastic MapReduce)... These experiments execute five distinct benchmarks: BenchClouds, Yahoo! YCSB, HI Bench, and TPC-W as listed in Table 1. The purpose is to check the performance of EC2 under different benchmark programs over varying data sizes.

The experimental setting applies a fixed instance type to scale out. For scale-up experiments, we have to change the instance types by program direction. Manual scaling is applied under program control in all experiments. Auto-scaling is not applied in scaling experiments on EC2 due to its brutal force provisioning policy. Some load-balancing was automatically practiced on the EC2 under the control of the EMR library.

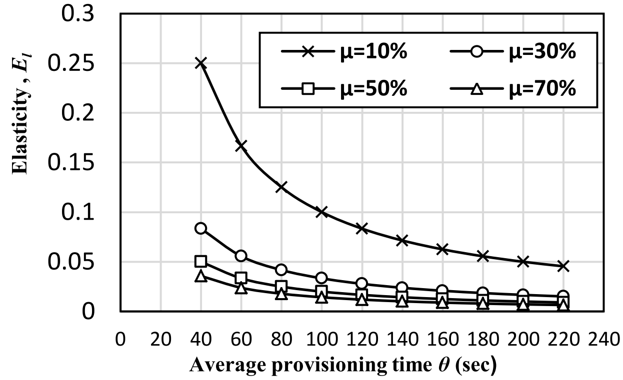


Fig. 4. The cloud elasticity plotted from Eq. (14).

7.1 Elasticity Effects in Scale-Out Strategy

We have conducted three scale-out benchmark experiments on EC 2 using the USC Benchcloud, HI Bench, and TPC-W, respectively in Figs. 5, 6, and 7.

7.1.1 Filtering of Twitter Spams on EC2

This is a benchmark testing the performance a mashup of two clouds (Twitter and EC2). In testing the BenchClouds benchmark, we scan through large amount of social media data (Tweets) collected from the Twitter cloud. Elastic MapReduce (EMR) software on EC2 is used to perform the fast spam filtering operations. The purpose is to filter out unwanted Spams from large Tweet dataset in a very short time [31].

In Fig. 5, we apply the *m1small* machine instance as listed in Table 3. This instance has a computing power of 1 ECU (Elastic compute unit) with 1 vCPU. The instance has 1.7 GB of RAM memory and 160 GB storage. Each instance is charged with \$0.044/hour with EMR surcharge applied. The data sets tested range from 10 GB to 100 GB and 1 TB.

For a small 10 GB dataset, there exists no apparent benefit by scaling out beyond eight instances (Fig. 12a). Efficiency drops sharply as the number of machine instances increase (Fig. 12c). The filtering process reaches the peak speedup with 128 nodes (Fig. 12b). For a large dataset of 1 TB, the execution time decreases by 58 times (Fig. 12a) with 128 nodes. Thus good speedup of 58 and 45 percent efficiency were achieved at 128 nodes (Fig. 12b, 12c). The small dataset shows poor productivity (such as 40 at 32 nodes in Fig. 12d), while the large dataset results in a peak productivity value at 32 nodes.

The scalability drops as we scale out from 2, 4, or 8 nodes up to 128 nodes (Fig. 5d) the drop in scalability (Fig. 5d) is closely correlated to the fast dropping in efficiency. On the other hand, the scalability in Fig. 5d varies closely with the change in productivity (Fig. 5c). The 1 TB curves (marked by a diamond legend) show that one can reach the peak productivity and thus peak scalability at 32 nodes.

7.1.2 HI Bench Results on Word Count

In HI Bench scale-out experiments, we increase the quantity of the same machine instances used. We plot the efficiency and productivity results in Fig. 6 by running the HiBench WordCount program using the EMR clusters up to 16 *m1*.

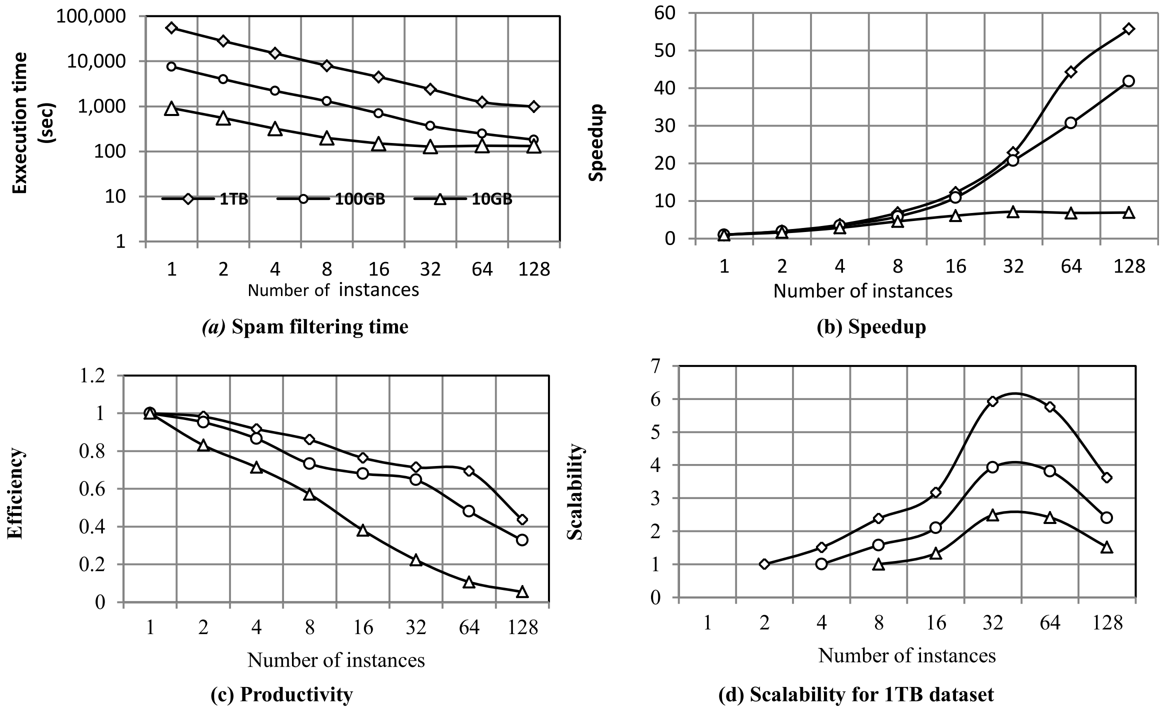


Fig. 5. Scale-out BenchClouds results on MapReduce filtering twitter spams over AWS EC2 of various sizes. Parts (a, b, c) apply the same legend. Part (d) shows the scalability measure from three initial machine instances.

small nodes. In general, the efficiency (Fig. 6a) decreases as we scale out to more nodes. However, for large data sets, the efficiency increases to a local peak at eight nodes and then it decreases slowly beyond eight nodes.

Depending on the data size, the productivity increases to different peak levels at different machine sizes. Foe example, the peak occurs at 12, 8 and 4 nodes for 10 GB, 5 GB and 1 G, respectively. After the peak, the productivity decreases more rapidly for small data size and slowly or flatly for larger data sizes. This trend is caused by the QoS and cost factors involved in Eq. (4). Other programs in HI Bench, such as Sort, can be also applied in the scaling experiments to be reported in Subsection 7.5.

7.1.3 TPC-W Scale-Out Results

This experiment is designed to test the TPC-W performance on EC2 under scale-out workload. The workload is generated by TPC client. We consider the workloads from 200 up to 2,400 users. In the scaling out process, we increase from 1, 4, 8 and 16 nodes up to 32 nodes. The m1.small instances are used in all scaling experiments. We report the throughput in WIPS (web interactions per section) and QoS measures in Figs. 7a, and 7b.

With small workloads (200 or 800 users), the WIPS count is rather flat after 4 nodes. The throughput reaches its peak of 340 WIPS at 12 nodes for 2,400 users. With 4,000 users, the peak value of 560 WIPS is reached at 20 nodes. The QoS reaches its peak value (100 percent) quickly after increasing the nodes to 4, 12 and 20, respectively (Fig. 7b). Fig. 7c shows the variation of productivity for different workloads. Again, the peak values occur at 4, 12 and 20 nodes for 800, 2,400 and 4,000 users, respectively.

The scalability plots in Fig. 7d start from 1, 4, 8 and 16 nodes. Due to two-order of magnitude difference of the 1-node curve (marked by x in Fig. 7d), we apply the wider scale on the left y-axis for this curve. The remaining three

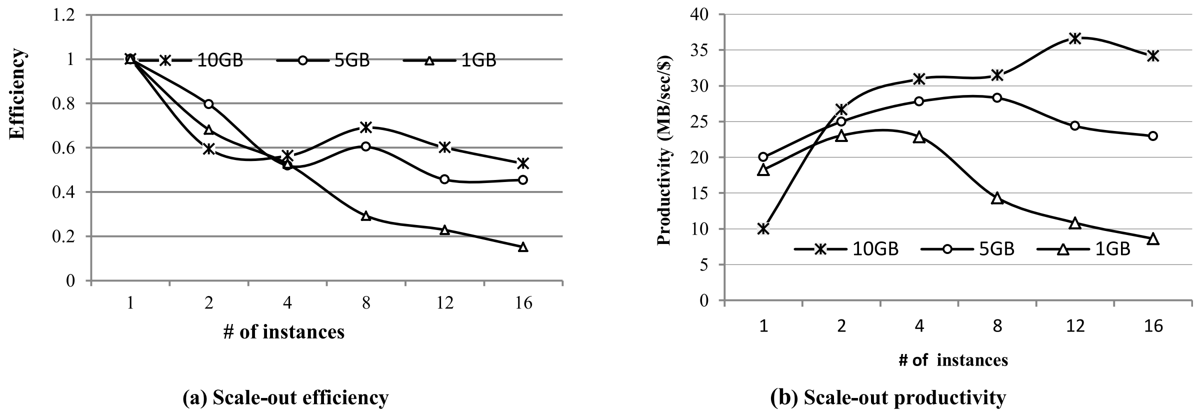


Fig. 6. Scale-out performance of HiBench on EC2 built with up to 16 *m1.small* machine instances. Three curves correspond to executing three workload sizes in the Word Count program.

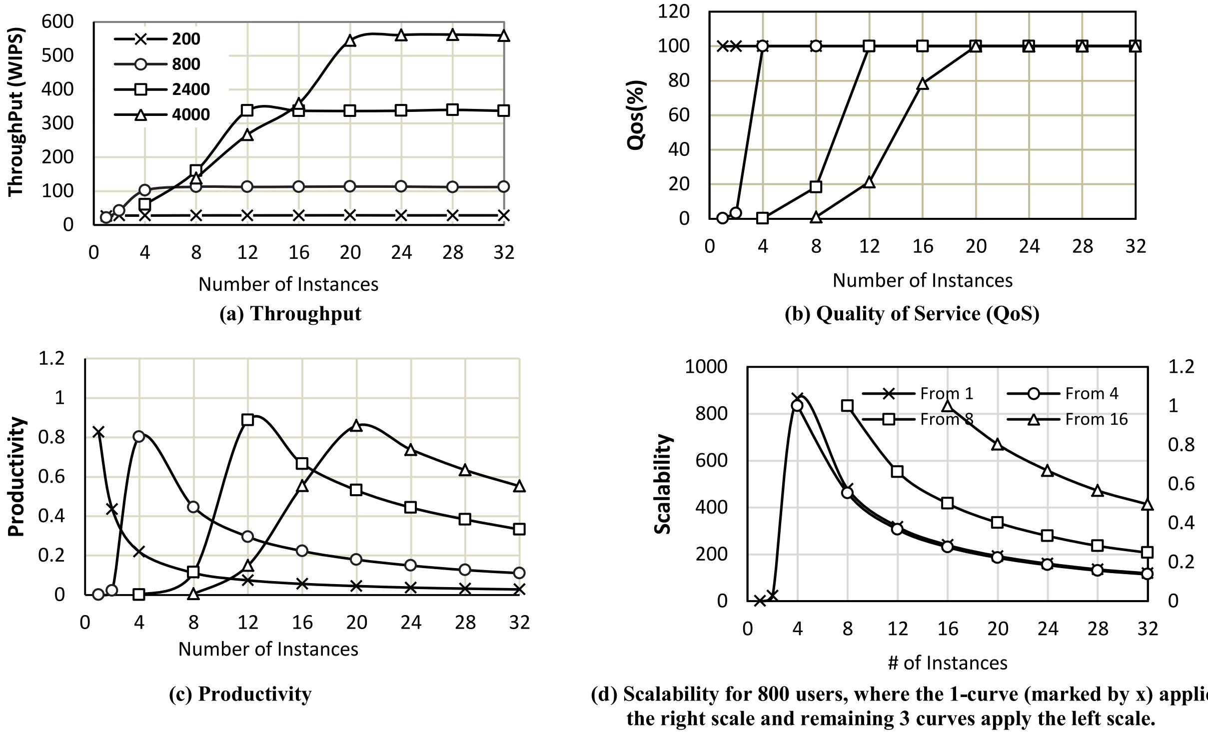


Fig. 7. Scale-out performance of TPC-W benchmark on Amazon EC2 cloud over increasing workload from 200 to 4,000 users. Parts (a, b, c) have the same legend. Part (d) scales from four initial machine instances.

curves are scaled by the right y-axis. Accordingly, the scalability with 800 users (Fig. 7d) has a sky rocket rise from two to four nodes. Similarly, we see the peak rises of p-Scalability at 4, 8 and 16 instances, respectively for more users. All scalability drops steadily after reaching their peaks.

7.2 Results of Scaling-Up Experiments

In scale-up experiments, we upgrade the machine instances from small to medium, large and extra-large types as given in Table 3 in order of increasing computing power (ECU and vCPU), memory and storage capacities. Of course, the renting cost increases from small to large, accordingly. Three scale-up experiments performed on the EC2 by running the YCSB, HI Bench, and TPC-W respectively.

In YCSB experiments, the EC2 system scales over 5 large or xlarge instances along the x-axis in Fig. 8. In TPC-W scale-up experiments, we follow the scaling sequence: *m1.small, m1.medium, m3.medium, m1.lrage,* and *m1.xlarge.* All scaling are done by program control in the experiments. Auto scaling cannot be implemented to automate the scaling-up process due to heavy overhead or low elasticity encountered.

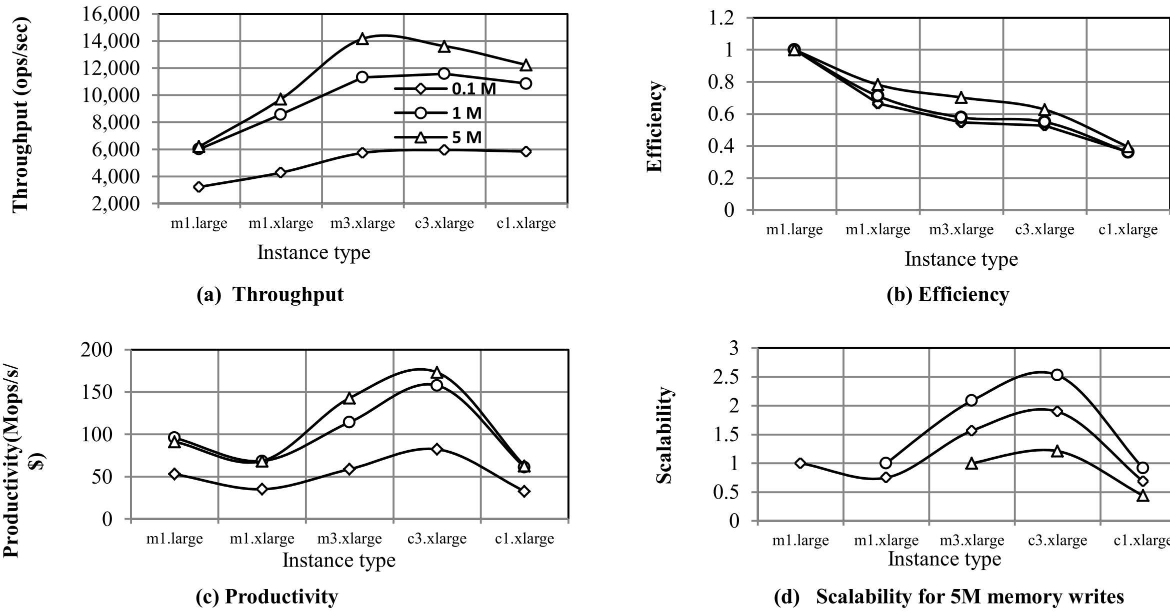


Fig. 8. Scale-up performance of Yahoo! YCSB on EC2 over increasing workload from 100 K to 5 M memory-access operations, where the same legend in Part (a) applies in all Parts. All instance types are specified in Table 3.

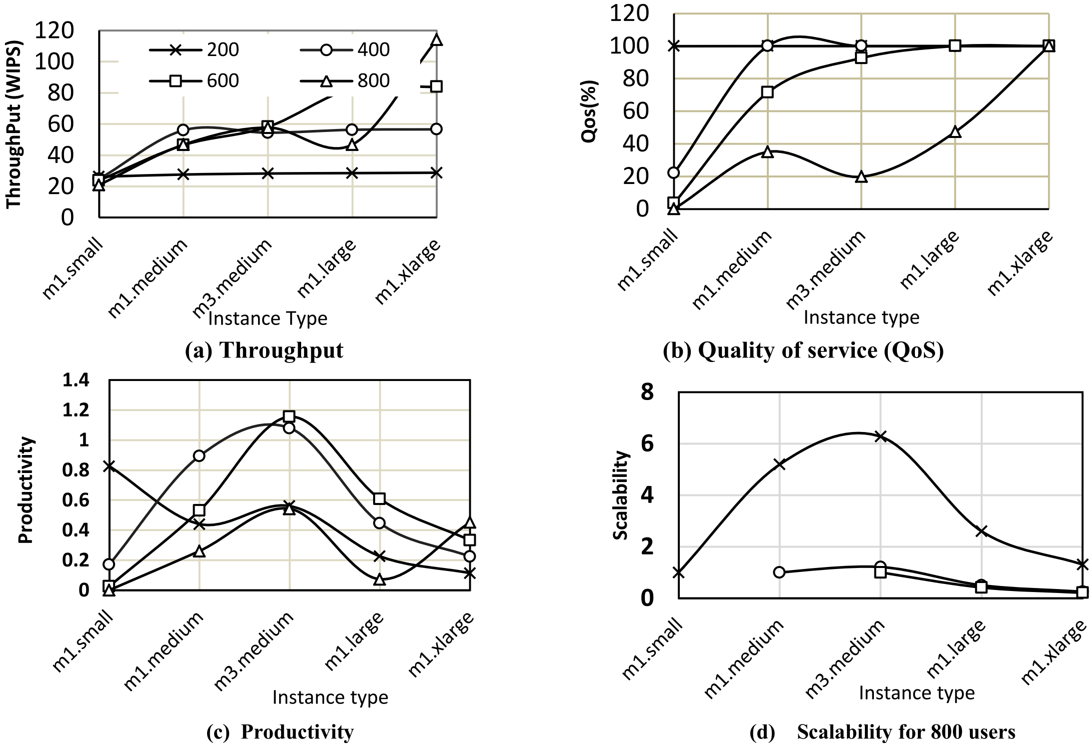


Fig. 9. Scale-up performance of TPC-W benchmark on Amazon EC2 clouds of various instant types over increasing workloads from 200 to 800 users.

7.2.1 Yahoo! YCSB Performance on EC2

We run the Yahoo! YCSB as part of the Cloudsuite data serving benchmark on AWS Hbase 0.92.0 cluster. We applied a write-intensive workload with 100 K and 5 M memory access operations on different types of instances. We use the default setting of Hbase. Figs. 8a, and 8b report the throughput and QoS, respectively. The cluster scales up to *m3.large* nodes.

Fig. 8a shows that for all three workloads, performance increases apparently when scaling up from m1.large to *m3.xlarge* instance, however for *c3.xlarge* and *c1.xlarge*, throughput and execution time almost remain the same as *m3.xlarge* instance. From Fig. 8b, the efficiency drops rapidly from *m1.large* to *m1.xlarge* and from *c3.xlarge* to *c1.xlarge*. This is due to the fact that scaling up does not catch the hardware resources increase.

We plot the productivity in Fig. 8c for 5 M memory operations. Here, we the set the QoS (cloud availability) to be 100 percent. As we scale up, the productivity reaches the peak values for all workloads at c3.xlarge. Fig. 8d is based on 5 M operations. The message being conveyed is that YCSB shows heavy memory-intensive database operations, and we can reach the highest productivity at c3.xlarge instance.

7.2.2 TPC-W Scale-Up Performance

We run the TPC-W benchmark with various workloads on five instance types with increasing computing power as seen in Table 3. The throughput increases with increasing workload in Fig. 9 except the 200-user curve is rather flat due to lack of work for scaling up to more powerful nodes. In Fig. 9b, the QoS for the 800-user curve is the low for smaller instance types due to overloading them. The QoS increases quickly to 100 percent.

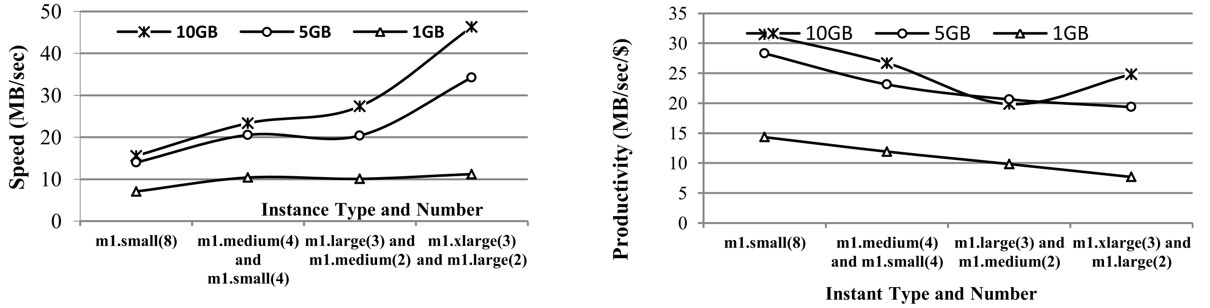
In Fig. 9c, we scale up from three node types under the workload of 800 users. Based on Eq. (4), we plot the productivity curves in Fig. 10d. The low value for 800-user curve is caused by its low QoS curve observed in Fig. 10b. All three curves reach the peak with the use of *m3.medium* node. We observe that with 800 or more users, the p-scalability reaches the peak with the *m1.medium* instance. After that, using more powerful nodes does not pay off. With even larger workload, say 4,000 users, the peak scalability may move further towards the right with larger instance nodes.

Note that the TCP-W results plotted in Fig. 9 have similar patterns as those YCSB results plotted in Fig. 7. However, they do differ in magnitude and peak performances. The main reason lies in different instance types used and different workloads applied. The operations counted in YCSB differ from the user count in TPC-W workload. They differ in about two orders in magnitude.

7.3 Mixed Scale-Up and Scale-Out Performance

For mixed scaling, four cluster configurations are specified along the x-axis in Fig. 10. The leftmost cluster has eight *small* instances with a total ECU count of 8. The next has four *medium* and four *small* instances with 12 ECUs. The next one has three *large* and two medium instances with 16 ECUs. The right cluster has three *xlarge* and two *large* instances with a total of 32 ECUs. Fig. 10 reports the HI Bench Word Count execution results.

Mixed strategy offers a wider range of ECU increase. The monotonic increase in speed (Fig. 10a) clearly supported this claim. For small data sizes (1 or 5 GB), the productivity (Fig. 8b) also decreases with large cluster used. For very large data set (10 GB), the productivity drops to a minimum point at the third large cluster and then increases again to a higher value for the rightmost cluster applied.



(a). Speed (throughput) (b) Productivity

Fig. 10. HiBench Word Count performance results on 4 EC2 clusters with mixed scale-up and scale-out nodes.

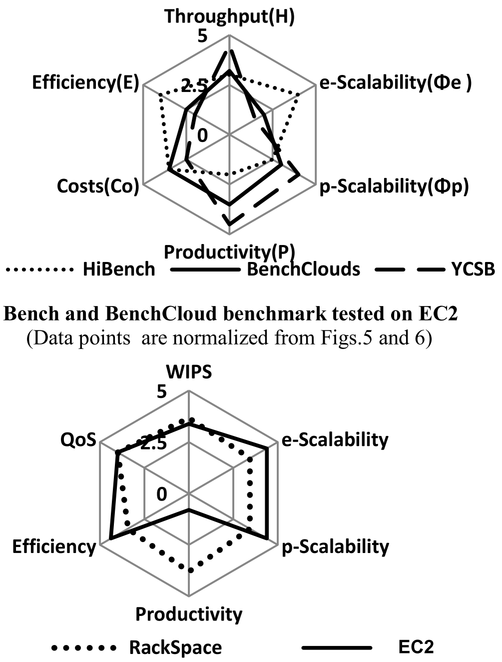
7.4 Effects of Changing Benchmarks or Cloud Platforms

Applying the relative performance models in Eqs. (11) and (12), we compare three benchmark programs: HiBbench, YCSB and BenchClouds and two cloud platforms: EC2 and Rackspace. These comparative studies reveal the strength and weakness in different benchmarks or cloud platforms.

7.4.1 HI Bench vs. BenchCloud Results

Fig. 11a compares the performance of three cloud benchmarks in six dimensions. The polygon data points are extracted and normalized from those in previous Figures. YCSB applies the scale-up workload with a higher throughput, productivity and scalability than the other two benchmarks.

The HighBench and BenchClouds apply the Elastic MapReduce resources with scale-out workload, they end up with comparable performance and higher cost than using the YCSB. HI Bench performs better in efficiency and e-scalability due to scaling out from a few to larger number of nodes. In general, we conclude that scaling-out should be practiced when the elasticity is high and scaling-up is in favor of using more powerful nodes with higher efficiency.



(a) HI Bench and BenchCloud benchmark tested on EC2 (Data points are normalized from Figs.5 and 6)

(b). EC2 and Rackspace tested with TPC-W

Fig. 11. Relative performance of compute clouds running different benchmarks in (a) and the same benchmark in (b).

7.4.2 TPC-W on EC2 vs. Rackspace Clouds

As plotted in Fig. 11b, we run the same TCP-W benchmark with 800 users on both EC2 and Rackspace platforms. The data of EC2 is extracted from Fig. 7. The Rackspace data are performed under similar workload and machine configurations. There is no performance difference in WIPS rate and QoS.

It is crucial to choose the proper set of performance metrics in cloud benchmarking experiments. From Figs. 2 and 10, we offered five different sets of performance metrics for modeling the IaaS, PaaS, SaaS, hybrid and mashup cloud configurations. Five benchmark programs are tested by which YCSB was embedded as part of the CloudSuite. These performance models can be modified to test other or new cloud benchmark suites as well.

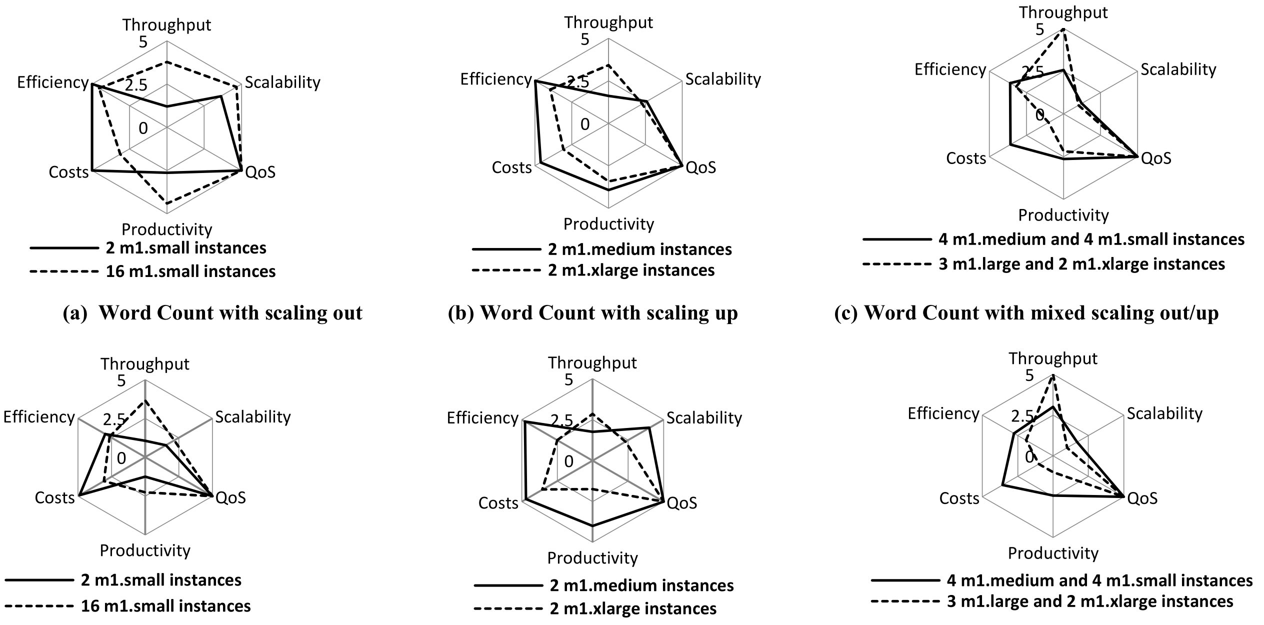
8 Elasticity Analysis of Scaling Strategies

Scaling out, scaling-up and mixed strategies are evaluated below. We compare their relative merits through executing two benchmark programs, Sort and Wordcount in HI Bench suite, on the AWS EC2 platform. The workload for these two programs has 10 GB of data elements. We measure the HI Bench performance of these two programs along six performance dimensions: *throughput, scalability, QoS, productivity, costs and efficiency*.

The QoS is mainly indicated by system availability which was recorded 99.95 $\sim$ 100 percent for all cluster configurations. Cost wise for the Word Count, the scale-out small cluster (solid polygons in Figs. 12a and 12d has the least service costs. The scale-up clusters in Figs. 12b and 12e cost more and the mixed cluster is the most expensive one to implement. Mixed scaling demands lot more considerations on tradeoffs between performance and cost incurred.

Speed wise, all mixed strategy for Sort (Figs. 12c and 12e) have the fastest throughput (or speed). The Word Count program shows slow throughput in all cases. The scale-up cluster shows very high efficiency for Word Count. The Sort clusters (dash-line polygons) show poor efficiency and throughput except high throughput for the mixed mode for sorting very large cluster in Fig. 12f.

In Fig. 12a, we see higher productivity for the large cluster (16 nodes) configuration. The peak values are application-dependent. Different benchmarks may lead to



(a) Word Count with scaling out (b) Word Count with scaling up (c) Word Count with mixed scaling out/up

(d) Sort with scaling out (e) Sort with scaling up (f) Sorting with mixed scaling out/up

Fig. 12. The performance maps of two HiBench programs on two EC2 cluster configurations for the scale-out, scale-up, and mixed scale-up/scale-out workloads over 10 GB of data elements.

different conclusions. In general, scaling-out should be practiced when the elasticity speed is high.

These performance maps are compared in Table 4 in terms of their polygon area values. Under each scaling case, we compare two cluster configurations. The polygon areas reported in Fig. 12 and Table 4 simply demonstrate a radar-chart method to compare the relative performance of testing various cluster configurations with a common benchmark.

In Table 5, we give a qualitative assessment of the three scaling techniques evaluated in HI Bench experiments on various EC2 configurations. The assessment is based on those quantitative measures reported in previous sections. We take a macroscopic view of the reported numerical results to reach some generalized observations on cloud performance under various operating constraints.

Over all, we find that scaling-out is the easiest one to implement on homogeneous clusters. The elasticity overhead is also lower in these cluster configurations. Scaling up is more complex to implement than scaling out due to the switching of node types. This will reduce the elasticity speed and prolong the reconfiguration overhead. The mixed scaling is the most difficult one to implement but offers the best flexibility to match with the workload change.

9 Conclusions and Suggestions

In general, the higher efficiency promotes the productivity, but the converse may not hold, necessarily. The QoS is based on user's objective. Different users may set their own satisfaction threshold for the QoS they can accept. The efficiency is controlled by the providers considering the interest of all user interests at the same time. We summarize below our major research findings from the comprehensive cloud benchmark experiments performed in 2014. Then, we suggest a few directions for further R/D in promoting cloud computing applications.

9.1 Summary of Benchmarking Findings

Over all, we find that scaling-out is the easiest one to implement on homogeneous clusters. The elasticity overhead is also lower in these cluster configurations. Scaling up is more complex to implement than scaling out due to the switching of node types. This will reduce the elasticity speed and prolong the reconfiguration overhead. The mixed scaling is the most difficult one to implement but offers the best flexibility to match with the workload change. Our research contributions are summarized below in 5 technical aspects:

TABLE 4
Performance Polygon Areas on Radar Charts in Fig. 12

Scale-Out Mode (Figs. 4a and 4d)	*Cluster Config.*	*2 small nodes*	*16 small nodes*
	Word Count	34.53	46.85
	Sort	17.02	23.65
Scale-Up Mode (Figs. 4b and 4e)	*Cluster Config.*	*2 medium nodes*	*2 xlarge nodes*
	Word Count	37.25	31.42
	Sort	41.84	21.22
Mixed Scaling Mode (Figs. 4c and 4f)	*Cluster Config.*	*4 medium and 4 small*	*3 large and 2 xlarge*
	Word Count	23.39	18.28
	Sort	22.81	11.90

TABLE 5
Assessment of Three Scaling Techniques based on HI Bench Benchmarking Findings on The EC2

Impact Factors	Scale-Out Technique	Scale-Up Technique	Mixed Scaling Technique
Elasticity speed, scaling complexity and overhead	Fast elasticity, possibly supported by auto-scaling and heuristics	High overhead to reconfigure and cannot support auto scaling	Most difficult to scale with wide range of machine instances
Effects on performance, efficiency, and scalability	Expect scalable performance if the application can exploit parallelism	Switching among heterogeneous nodes may reduce scalability	Flexible app, low efficiency, and resource utilization
QoS, costs, fault recovery, and cloud productivity	Cost the least, Easy to recover, Incremental productivity	More cost-effective, but Reduced QoS may weaken the productivity	High costs, difficult to recover, expect the highest productivity

1) New performance metrics and benchmarking models are proposed and tested in cloud benchmark experiments. We study the scalability performances driven by efficiency and productivity, separately. This approach appeals to different user groups with diversified performance demands.
2) Sustained performance of clouds comes mainly from fast elastic resources provisioning to match with the workload variation. Scaling-out should be practiced when the elasticity is high, Scaling-up is in favor of using more powerful nodes with higher efficiency and productivity.
3) To provide productive services, both scale-up and scale-out schemes could be practiced. Scale-out reconfiguration has lower overhead to implement than those for scaly-up scheme. The elasticity speed plays a vital role in minimizing the over- or under-provisioning gaps.
4) We show that scaling up is more cost-effective with higher productivity and p-scalability in YCSB and TPC-W experiments. These findings may be useful to predict other benchmark performance if they attempt to scale out or scale-up with similar cloud setting and workload.
5) The cloud productivity is greatly attributed to system elasticity, efficiency, and scalability, all affecting performance. The cloud providers must enforce performance isolation for quota-abiding users at the expense of quota-violating users.

9.2 Suggestions for Further Work

Three suggestions are made below for further work. The ultimate goal is to generate commonly accepted cloud benchmarks and testing techniques. These tasks are naturally extendable from the cloud performance models being proposed.

6) Other cloud benchmarks: CloudStone [35] CloudCmp [27], and C-meter [37], could be also tested with the new performance models presented. Future benchmarks are encouraged to evaluate PaaS and SaaS clouds.
7) To make clouds universally acceptable, we encourage cloud researchers and developers to work jointly in developing a set of application-specific benchmarks for important cloud and big-data application domains.
8) The cloud community is short of benchmarks to test cloud capability in big-data analytics and machine learning intelligence. This area is widely open, waiting for major research/development challenges.

Acknowledgments

This work was supported in parts by National Basic Research (973) Program of China under the Grant No. 2011CB302505, by the High-Tech Research and Development (863) Programs of China under Grant No. 2012AA012600, by National Science Foundation of China under Grants Nos. 61472196, 61433008 and 61073003, and by the University of Southern California. The work of Kai Hwang was supported by EMC Co. and Intellectual Ventures Inc. Kai Hwang is the corresponding author of the article.

References

[1] R. Appuswamy, C. Gkantsidis, D. Narayanan, O. Hodson, and A. Rowstron, "Scale-up vs scale-out for hadoop: Time to rethink," presented at the *ACM Symp. Cloud Comput.*, Santa Clara, CA, USA, Oct. 2013.
[2] X. Bai, Y. Wang, G. Dai, W. T. Tsai, and Y. Chen, "A framework for collaborative verification and validation of web services," *Component-Based Software Engineering*, Springer, 2007.
[3] C. Binnig, D. Kossmann, T. Kraska, and S. Loesing, "How is the weather tomorrow? Towards a benchmark for the cloud," in *Proc. ACM 2nd Int. Workshop Testing Database Syst.*, Jun. 2009.
[4] Bitcurrent, Inc. (2010). Cloud computing performance report. [Online]. Available: http:// bitcurrent.com
[5] A. Bondi, "Characteristics of scalability and their impact on performance," in *Proc. 2nd Int. Workshop Softw. Performance*, 2000, pp. 195–203.
[6] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, Hype, and Reality for delivering computing as the 5th utility," *Future Gen. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, 2009.
[7] J. Cao, K. Hwang, K. Li, and A. Zomaya, "Optimal multiserver configuration for profit maximization in cloud computing," *IEEE Trans. Parallel, Distrib. Syst.*, vol. 24, no. 6, pp. 1087–1096, Jun. 2013.
[8] G. Chen, X. Bai, X. Huang, M. Li, and L. Zhou, "Evaluating services on the cloud using ontology QoS model," in *Proc. IEEE Int. Symp. Service Oriented Syst. Eng.*, 2011, pp. 312–317.
[9] Y. Chen, A. Ganapathi, R. Griffith, and R. Katz, "The case for evaluating mapreduce performance using workload suites," in *Proc. IEEE Int. Symp. Model., Anal. Simulation Comput. Telecom Syst.*, 2011 pp. 390–399.
[10] CloudHarmony. (2014). Benchmark evaluation of 114 public clouds. [Online]. Available: http://cloudharmony.com/clouds
[11] B. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," in *Proc. 1st ACM Symp. Cloud Comput.*, 2010, pp. 143–154.
[12] J. Dongarra, J. Martin, and Worlton, "Computer benchmarking: Paths and pitfalls," *IEEE Spectrum*, vol. 24, no. 7, pp. 38–43, Jul. 1986.
[13] M. Farber and S. Kounev, "Existing cloud benchmark efforts and proposed next steps," Slide Presentation, Karlsruhe Inst.Technol. (KIT), Karlsruhe, Germany, Aug. 31, 2011.
[14] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Clearing the clouds: A study of emerging scale-out workloads on modern hardware," presented at the *ACM 17th Int. Conf. Architectural Support Program. Languages Operating Syst.*, London, United Kingdom, Mar. 2012.

[15] E. Folkerts, A. Alexander, K. Sacks, A. Iosup, Markl, and C. Tosun, "Benchmarking in the cloud: What it should, can and cannot be," presented at the 4th TPC Technol. Conf. Performance Eval. Benchmarking, Istanbul, Turkey, Aug. 2012.
[16] J. Gao, X. Bai, and W. T. Tsai, "Cloud-testing: Issues, challenges, needs and practice," *Int. J. Softw. Eng.*, vol. 1, no. 1, pp. 9–23, 2011.
[17] A. Gupta and V. Kumar, "Performance properties of large scale parallel systems," in *Proc. 26th Hawaii Int. Conf. Syst. Sci.*, 1993, pp. 234–244.
[18] N. Herbst, S. Kounev, and R. Reussner, "Elasticity in cloud computing: What it is, and what it is not," presented at the *Int. Conf. Autonomic Comput.*, San Jose, CA, USA, Jun. 2013.
[19] M. Hill "What is scalability ?," *ACM SIGARCH Comput. Archit. News*, 1990.
[20] S. Huang, J. Huang, J. Dai, and T. Xie, and B. Hong, "The hi-bench benchmark suite: Characterization of the mapreduce-based data analysis," in *Proc. Int. Conf. Data Eng. Workshops*, Mar. 2010, pp. 41–51.
[21] K. Hwang, G. Fox, and J. Dongarra, *Distributed and Cloud Computing*, Morgan Kaufmann Publisher, 2012.
[22] K. Hwang and Z. Xu, *Scalable Parallel Computing*, Chapter 2 on Performance Benchmarking, McGraw-Hill, 1998.
[23] K. Hwang, Y. Shi, and X. Bai, "Scale-out and scale-up techniques for cloud performance and productivity," presented at the *IEEE Cloud Comput. Sci., Technol. Appl. (CloudCom)*, Singapore, Dec. 18, 2014.
[24] A. Iosup, S. Ostermann, M. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "Performance Analysis of cloud computing services for many-tasks scientific computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 6, pp. 931–945, 2011.
[25] R. Krebs, C. Momm, and S. Knounev, "Metrics and techniques for quantifying performance isolation in cloud environments," presented at the *ACM QoSA*, Bertino, Italy, Jun. 2012.
[26] A. Li, X. Yang, S. Kandula, and M. Zhang, "CloudCmp: Comparing public cloud providers," in *Proc. 10th Annu. Conf. Internet Meas.*, Nov. 2010, pp. 1–14.
[27] Z. Li, L. O'Brien, H. Zhang, and R. Cai, "On a catalogue of metrics for evaluating commercial cloud services," in *Proc. ACM/IEEE 13th Int. Conf. Grid Comput.*, Sep. 2012, pp. 167–173.
[28] P. Mell and T. Grance, "The NIST definition of cloud computing," *NIST Special Pub.*, vol. 800, no. 145, Jul. 2011
[29] M. Michael, J. Moreira, D. Shiloach, and R. Wisniewski, "Scale-up x scale-out : A case study using nutch/luene," in *Proc. IEEE Int. Parallel Distrib. Proc. Symp.*, Mar. 26, 2007, pp. 1–8.
[30] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and, D. Epema, "A performance analysis of EC2 Cloud Computing services for Scientific computing," in *Proceedings of the International Conference On Cloud Computing*, Springer, 2010.
[31] C. Pham "QoS for cloud computing," Slide presentation, Univ. Pau, Pau, France, May 10, 2011.
[32] D. Plummer, D. M. Smith, T. J. Bittman, D. W. Cearley, D. J. Cappuccio, D. Scott, R. Kumar, and B. Robertson. (2009). Five refining attributes of public and private cloud computing. Gartner Research. [Online]. Available: http://gartner.com/DisplayDocument?doc-cd=167182
[33] U Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "A cost-aware elasticity provisioning system for the cloud," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, Jun. 2011, pp. 559–570.
[34] W. Smith, "TCP-W: Benchmarking: An E-commerce solution," Intel, 2005.
[35] W. Sobel, S. Subramanyam, A. Sucharitakul, J. Nguyen, H. Wong, Patil, and D. Patterson, "Cloudstone: multi-platform, multi-language benchmark and measurement tools for web 2.0," in *Proc. 1st Workshop Cloud Comput. Appl.*, Oct. 2008.
[36] W. Tsai, Y. Huang, and Q. Shao, "Testing the scalability of SaaS applications," in *Proc. IEEE Int. Conf. Service-Oriented Comput. Appl.*, 2011, pp. 1–4.
[37] N. Yigitbasi, A. Iosup, D. Epema, and S. Ostermann, "C-Meter: A framework for performance analysis of computing clouds," in *Proc. IEEE/ACM Proc. 9th Int. Symp. Cluster Comput. Grid*, 2009, pp. 472–477.

Kai Hwang received the PhD degree from the University of California at Berkeley, Berkeley, CA. He is currently a professor of electrical engineering and computer science at the University of Southern California, Los Angeles, CA. He has published eight books and 245 papers in the areas of computer architecture, parallel processing, network security, P2P/Grid and cloud computing. The IEEE CloudCom 2012 has extended him a Life-Time Achievement Award for his pioneering contributions in parallel and distributed computing. He is a life fellow of the IEEE.

Xiaoying Bai received the PhD degree in computer science from Arizona State University, Tempe, AZ. She is currently an associate professor of computer science and technology at Tsinghua University, Beijing, China. She specializes in software engineering, web services, and cloud computing. She has published 90 research papers. She is a member of the IEEE.

Yue Shi received the BS degree from Huazhong University of Science and Technology, Wuhan, China, and the MS degree from the University of Southern California, Los Angeles, CA. She is currently working toward the PhD degree in electrical engineering at the University of Southern California. Her research interests include big data analytics, mobile and cloud systems, social networks, and mashup applications.

Muyang Li received the BS degree in computer science and technology from Tsinghua University, Beijing, China, in 2013. He is currently working toward the MS degree in computer science from Tsinghua University. He has accepted to join Google in 2015.

Wen-Guang Chen received the BS and PhD degrees in computer science from Tsinghua University, Beijing, China, in 1995 and 2000, respectively, where he is currently a professor of computer science and Technology. His research interest covers the areas of parallel and distributed computing, programming model, cloud computing, and performance evaluation.

Yongwei Wu received the PhD degree from the Chinese Academy of Sciences, Beijing, China, in 2002. Currently, he is a professor of computer science and technology at Tsinghua University, Beijing, China. His research interests include parallel and distributed processing, distributed file system, and cloud storage. He is a member of the IEEE.