J Grid Computing (2017) 15:557–572
https://doi.org/10.1007/s10723-017-9412-4

Cost-Performance Modeling with Automated Benchmarking on Elastic Computing Clouds

Hongyan Mao · Zhengwei Qi · Jiangang Duan · Xinni Ge

Received: 6 June 2016 / Accepted: 5 October 2017 / Published online: 23 October 2017

Abstract The importance of evaluating performance of cloud systems has been increasing with the rapid growing market demands for cloud computing. However, the performance testers often have to go through the hassle of tedious manual operations when interacting with the cloud. A cloud performance evaluation framework is designed for both broad cloud support and good workload extensibility, which provides an automatic interface to monitor the capability and scalability of Infrastructure-as-a-Service cloud systems. Cloud API modules are implemented for Amazon EC2 service and OpenStack. It can achieve flexible control workflows for multiple of different workloads and user customization to test scenarios. With several built-in workloads and metric aggregation methods, a series of tests is performed on our private clouds to compare the performance and scalability from multiple aspects. A methodology is also proposed to build a cost-performance model to better understand and analyze the efficiency of different types of cloud systems. Based on the results of the experiments, the model indicates a polynomial relation between performance per instance and the overall cost.

Keywords Cloud computing · Performance analysis · Measurement · Performance modeling

H. Mao (✉)
School of Computer Science and Software Engineering, Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China
e-mail: hymao@sei.ecnu.edu.cn

Z. Qi · X. Ge
School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China
e-mail: qizhwei@sjtu.edu.cn

X. Ge
e-mail: gexinni@sjtu.edu.cn

J. Duan
Intel APAC R& D Center, Shanghai, China
e-mail: jiangang.duan@intel.com

1 Introduction

Cloud Computing has been one of the most important and popular topics in the computer industry [1, 2]. Cloud systems process massive amounts of data with a number of distributed servers and devices, and provide flexible and on-demand basis services. It promises high availability, scalability and reliability, which is expected as the key features of modern industrial systems.

Among the major models of cloud computing service, such as Software-as-a-Service (SaaS) [3], Platform-as-a-Service (PaaS) [4], Data-as-a-Service [5], Network-as-a-Service [6] and Benchmarking-as-a-Service [7], Infrastructure as a Service (IaaS) [8] aims at providing lower level resources, often in the format of virtual machines, for the customers and charging in a pay-per-use basis manner.

A lot of projects and products are developed aiming at creating a cloud system with high availability and

scalability. Amazon [9], Microsoft [10], HP [11] and many other enterprizes sell their public cloud service to individuals and small companies. Besides, there are many open-source or free cloud computing software, such as Apache CloudStack [12], Opennebula [13], Eucalyptus [14]. OpenStack [15] is an open-source project for building private and public clouds and has great potential with more than 200 companies participating in.

When a customer decides to move to a cloud, there are many alternatives to choose. The customer can purchase some resources from a cloud provider or build his/her own private cloud with one of the cloud projects. To compare different choices, a good cloud service benchmark becomes valuable for the cloud customers. On the other hand, cloud providers also pay high attention on the cloud evaluation methods to prove their own advantage against other competitors. Massive kinds of tests need to be performed on the system with different configurations.

However, it is usually quite difficult to evaluate a cloud due to the complexity of systems. In real-world practice, there are always different types of workloads which assess different aspects of the system and explore a large scale of the parameter space of the workload combinations. Cloud customers or providers have to spend a lot of time and efforts in preparing the environment and configuring scripts. Additionally, cloud computing brings more important performance goals such as scalability, availability and quality of services, which ask for more comprehensive and realistic test scenarios. What is more, a long testing period increases the possibility of encountering system errors, which may fail the test.

There have been quite a lot of tools assessing and comparing performance on different cloud systems [16, 17] but they either provide only a limited variety of clouds and workload parameter space, or need a quantity of human involvement during the whole test period, including environment preparation and final results analysis.

Considering the problems above, a benchmark framework is designed, which consists of an automatic performance evaluation tool working on different types of cloud and a flexible framework to expand the workload set, as an IaaS cloud performance benchmark leveraging workload models to free some configuration labors on testers and keep the simplicity and flexibility of testing. CloudScore recognizes two different kinds of target testers. One of them is likely to be cloud providers who require to add and configure different kinds of workloads to explore the hidden potential of their clouds. The other kind of testers, likely to be service customers, only cares about the performance reports from different clouds of their own interests. CloudScore provides a framework for the former testers to develop their set of workloads with different scenarios and let the latter take the advantage of these test cases to focus on their tests comparing among clouds and configurations.

One important design principle of CloudScore is to manage resources on the cloud automatically. It works with the cooperation of the cloud provider's interfaces by abstracting the cloud as Cloud System Under Test (Cloud SUT). Particularly the resources are Virtual Machines (instances), IP addresses, virtual storage (volumes), images and so on. The workload is composed with a coordinator, a configuration file and some images holding the testing environment. The tester of CloudScore may use a web-based GUI to give the test plan, called as a Project, including the Cloud SUTs and the selected Workloads, as well as the provision mode. Basically, there aretwo modes of provision, the parallel mode and the sequence mode. The former launches all required resources and starts to run at the same time, while the latter continues a dynamic provision-then-run process during a given period.

The benchmark framework about resource management is introduced in [18]. Based on this research, a cost-performance benchmark framework is proposed which provides flexible cloud support expansions and extensible workload control methods. In our CloudScore, Cloud API modules are implemented for Amazon EC2 service and OpenStack. A series of experiments are carried out on our private clouds to show the robustness and compatibility of our framework by comparing different cloud platform. A cost-performance modeling is introduced to analyze the cost-performance relations of the cloud system to evaluate the cost efficiency of resource utility.

The rest of the paper is organized as follows. Section 2 mentions some related efforts about cloud computing and cost performance. And then Section 3 discusses the challenges of evaluating the performance of a cloud system due to the complicate characteristics. Some of the important design targets of the CloudScore are introduced in Section 3 and the modeling method in Section 4. More details about

implementation is presented in Section 5. A series of experiments are carried out on OpenStack private clouds. The analysis results are presented in Section 5. Section 6 includes a brief conclusion about the whole paper, as well as our expectations of the future work.

2 Related Work

There have been quite a lot of tools to access the performance of virtualized clusters or web services. vConsolidate [19] is a workload methodology introduced by Intel, and is intended to characterize the performance of virtualization clusters. VMmark [20] is a benchmark designed for the scalability of virtualized system released by VMware. These works focus on virtual clusters and do not consider the software layer in the scope of cloud management. But CloudScore provides the abstract of cloud management operations. It simplifies the automatic allocation of cloud resources to reduce the workload of the test staff.

Plenty of works concentrating on cloud computing have sprung up in recent years. CloudSim [19] models the behavior of cloud system components and introduces generic provisioning techniques for applications with good extensibility. [20] presents a characterization of Dropbox and shows possible bottleneck in the current system architecture and storage protocol. The paper introduces a credit model for the resource usage providing automatic management and presents a model prototype in federated cloud [21]. In this paper, a new method is designed in CloudScore to illustrate the performance data relations of the cloud system obtained from single load or multi load operations, and it provides the changing models of the performance with different scales of the workload. Utilizing the models, cloud providers can optimize allocation of cloud resources and service customers can compare among clouds with a better choice.

Works concentrate on some aspects of distributed systems and cloud services, rather than evaluating and profiling of IaaS cloud system itself. Cloud Serving Benchmark is a generic open-source tool focused on assessing cloud serving, key-value storage and some other NoSQL services. [22] proposes a mixed approach for cloud service negotiation to fulfill the Service Level Agreement of cloud providers and cloud consumers. However, it is quite complex and time-consuming to set up the test environment and configuration. CloudStone consists of an open-source Web 2.0 social application, and is available for automatic load generation for testing performance in different deployment environments. [23] develops a novel parallel intelligent algorithm to solve the large scale service composition optimal-selection problem with numerous constraints in Cloud manufacturing. BigDataBench [24] provides diverse and representative data sets, together with broad application scenarios to benchmark web-based big data systems and architectures. [25] creates a model with quality dimensions and metrics that targets general cloud services, containing six quality dimensions to represent, measure, and compare clouds and build a mutual understanding among different cloud providers. In [26], they investigates the applications of Internet of things technologies in cloud manufacturing and designs a five-layered structure resource intelligent perception and access system.

Public cloud systems also attracts a lot of attention. The paper gives a federated hybrid cloud model of Infrastructure as a Service (IaaS) to provide eScience and depicts an architecture for constructing Platform as a Service (PaaS) and Software as a Service (SaaS) [27]. CloudHarmony [28] provides a form of Benchmark-as-a-Service. With its online interface, the tester is able to generate reports on performance comparison across public clouds. CloudCmp [29] is a multi-cloud comparison framework with micro-benchmark-level profile of multiple public cloud providers. The works mentioned above focus on assessing public clouds services, but not available for evaluating private clouds. Besides, some efforts have been made on analyzing the architecture of cloud systems instead of comparing among different clouds. DeepDiv [30] addresses some performance issues of interference between instances co-located on the same physical machine in IaaS cloud systems.

M. Caballer, et.al presents a component model in order to reduce the access and the usability of IaaS clouds by automating the VMI selection, deployment, configuration, installation of Virtual Appliances. CloudSuite [31] provides a benchmark that performs scale-out workloads testing and can be used to address the inefficiencies in the modern micro-architecture. CloudGauge [32] is proposed as a dynamic and an experimental benchmark for virtualized and cloud environments, which attempts to provide abstractions

at the workload level for supporting more complex workload interactions. They use some pre-defined workloads without good workload extensibility and only perform on some specific cloud platforms. But, in CloudScore, it uses the unitization and abstract of workload structure and workload running logic to improve the reuse rate of operation logic and the scalability of load set. Utilizing the management module of cloud operation, CloudScore can interface with the specified cloud system expediently. And the encapsulation of complex resource management operation greatly reduces the interaction between the tester and the tested cloud system.

IBM proposes an open-source project CloudBench [33] as a framework that perform cloud-scale evaluation and benchmarking automatically by running controlled experiments. C-Meter [17] is expected as a portable and extensible framework for generating and submitting workloads on Amazon EC2 Cloud. However, they are not easy for configuring a new workload or customizing a metrics-calculation method to profile the system, and do not achieve multiple cloud systems support and flexible application extension at the same time.

And there are increasing number of projects focused on OpenStack performance testing. OpenStack itself proposes project Rally [34], a Benchmark-as-a-Service project for evaluating its sub-projects. It is expected to perform the specific, complex and reproducible tests under real deployment scenarios, but is still on early stage of developing.

Third parties also make a lot of contribution to OpenStack deployment and evaluation. Fuel [35] of Mirantis provides a broad suite of tests to validate all key services are running correctly, in addition to some real-world functional tests that aims at isolating problems in specific OpenStack subsystems.

Two basic kinds of workflow to control the execution of workload set is designed in CloudScore. Parallel mode is more accurate to control the parallel degree, gives a constant and diverse pressure on various components of the cloud system, and imitates the mixed behavior of different user applications. Sequence mode gives an incremental load to the cloud in the beginning and may keep a dynamic balanced period if the whole running period is long enough which contains some uncertainty factors to make the test more realistic to the real-world scenarios.

We introduce a new tool which is convenient to expand the workload set and customize the execution workflow, which supports multiple of IaaS cloud platforms for comparison.

3 Design

CloudScore is defined as a benchmark for evaluating the performance of IaaS clouds and is expected to provide flexible workload extensibility and automatic execution, together with result analysis and aggregation methods. It runs the Workloads on the Cloud SUTs and reports the results after analyzing the performance data. CloudSUT specifies the access method of the cloud, together with the abstraction of available resources. Particularly, a Cloud SUT contains the details about access authentications and resources identification of the particular public or private cloud system. A Workload is composed of a set of packages of executable binaries and scripts to generate load on the system, as well as the result computing methods. Several Workloads can be arranged into a Workload Set flexibly and extensively to perform tests in a more complex way.

Figure 1 illustrates a high level view of CloudScore. After the tester submits the Project, Core Daemon parses the user input and generates the corresponding Job. Core Daemon transfers the user submission to jobs and process them in a First-In-First-Out manner. Core Daemon maintains a First-In-First-Out (FIFO) queue to hold all the Jobs. The jobs will be processed in a First-In-First-Out manner. When a Job starts, a Coordinator Daemon is generated to control the execution beginning with launching instances and resources. Coordinator Daemon operates the cloud

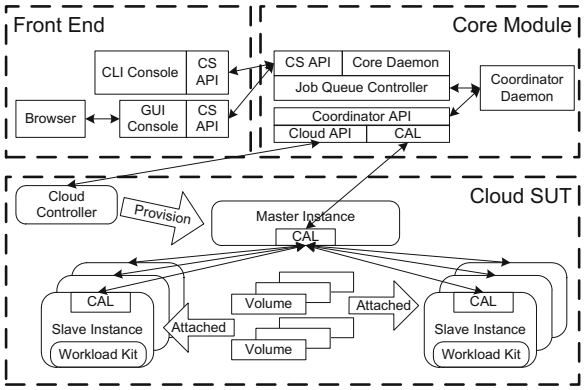

Fig. 1 Overall view of main components in CloudScore

with the assistance from Cloud API, which abstracts APIs exposed by the cloud controller. When an instance is launched and becomes available with all required resources ready to use, it waits for the start signal to begin the running process. In general, for every Cloud SUT, CloudScore boots a Master instance, as the proxy of all the test instances under test (slave instances). Coordinator Daemon directly communicates with instances in the cloud via the Communication Abstract Layer (CAL) and informs the test instances to start the run after all required resources are allocated and associated. The workload running period can be easily customized and by default is divided into four stages: Prepare, Run, Stop and Post.

The original results will be simply processed and sent back to Coordinator Daemon for more complicated calculations. During the whole job running period, Core Daemon tracks all the status and results information about the Job and generates persistent archives. The front end uses the Django as the development framework deployed in Apache Server and some configure operations are listed in Table 1.

From the point of view of a performance tester, Cloud SUT is treated as a block box and does not need to know more details about the cloud than a normal cloud user. CloudScore asks the user for the necessary authentication with access information, and takes the responsibility of sending requests to the cloud under most situations. It ensures the tester to focus on performance evaluation and analysis, rather than suffer from a lot of cumbersome commands to operate on the cloud resources.

Table 1 Configure operation

Operation	Arguments	Descriptions
list_cloudtypes	–	Cloud types supported
get_cloudtype	cloud type	Obtaining cloud module
get_clouddetails	cloud type	Obtaining cloud resources
list_workloads	–	Workload sets supported
get_workload	workload name	Obtaining detailed workload information

3.1 CloudScore Workload

The traditional benchmarks, which are often used to evaluate one or several aspects such as computation capacity or network service of application system, are not appropriate for assessing a cloud system, since the testing environment becomes quite complicated in an IaaS cloud. CloudScore provides a set of practical Workloads supporting automatically evaluating the performance on different aspects of a cloud system, and it is also pleasant to receive new workloads from a senior tester, or so called Workload Developer. A CloudScore Workload specifies several roles that take different responsibilities. For example, a Workload assessing networking may have two Roles, one for sending packets and the other for receiving. To enable a Workload, only three components are necessary, one configuration file in XML format, one coordinator script defining its processing workflow, and one or several images supplying the environment. And the coordinator is reusable and the configuration file is easy to be modified for new parameters. The configuration file defines the basic information of the Workload including the necessary resources, the parameters exposed to the tester, and the specific run stages of all roles.

The coordinator script controls the overall steps to run the Workload involved with provision and un-provision parts. CloudScore provides a set of Coordinator APIs for the coordinator script. The default coordinator script expects to be enough for most of the simple workload and if more complex workflow controlling logic is required, it is also easy to override it with another script.

The result processing script gives a metric processing method, which filters and calculates all the performance data retrieved from the instances and aggregates to a final score after analyzing all sub-metrics. The result process is isolated from the entire workflow of a Workload considering of the variety of different requirements to process results of different workloads.

Creating a new Workload is convenient by reusing the default coordinator script. It is also quite easy for the tester to configure the own workflow with well-defined Coordinator APIs. The developer can supply their Workload kits in some images rather than a single compressed package, which allows to hold all necessary dependency packages, special configurations and environment variables.

3.2 CloudScore Workload Set

To simulate the realistic usage scenario of cloud systems, it needs to involve the combination of several

workloads and the complex execution process running for a long period. We introduce the concept of Workload Set to allow more complicated methods for controlling the execution of several workloads at the same time.

A performance tester submits a test Project that specifies the particular Cloud SUTs and Workload or Workload Set to CloudScore. Then CloudScore generates a corresponding Job and add it to the Job Queue. Job Queue Controller is the control center of CloudScore. All Jobs in the queue are pending for execution in a FIFO way. At one time there will be at most one Job under the state of running. Every Job will be allocated with a Coordinator Daemon to control its execution. When a Job starts to run, the Coordinator Daemon will begin to provision the Master instance and Slave instances. The Slave instances are responsible for running the workloads while the Master instance is responsible for forwarding requests between the Coordinator daemon and Slave instances. This design asks for only one extra public IP for one test process and tries to reduce the interference the cloud as much as possible. CloudScore assumes the influence of the existence of one master instance to cloud performance could be ignored since a cloud is supposed to provide a large provision capacity.

After a given period of running, CloudScore will terminate the run, process the raw data and clean up the cloud environment by releasing all allocated resources. Adding a new Workload Set is not more complicated than adding a workload, since all the roles and images of the workloads are ready to use. The tester only needs to write a new coordinator script for the wanted scenario and specify the method to handle the results collected from different workloads.

3.3 Multi-workload Scenarios

Figure 1 describes the overall workflow of processing a job. When the tester commits a test plan (Project) to the Job Queue, the job will enter the pending stage. When the Job starts to run, it creates a Coordinator Daemon and the daemon loops to execute for every Workload. Each Workload first provisions and allocates all necessary resources. Then it turns into the workload execution period. The workflow divides the whole running process inside the instances into four stages by default.

First, on Prepare Stage, CloudScore prepares and more importantly, verifies all the parameters received from the tester, together with the whole environment, to make sure the run is supposed to be successful. It also provides the logics to decide some parameters dynamically.

Second, on Run Stage, it starts the workload and will not stop until the termination condition is satisfied or time is up. On Stop Stage, it cleans the environment by terminating processes and services, deleting files generated during the run and resetting some environment parameters. And on Post Stage it collects raw data to generate a result package. The last stage is also responsible for the first-round data filtering and processing. It is easy to add, remove or customize the stages with the relevant parameters by modifying the configuration file of the particular workload.

Two basic kinds of workflow to control the execution of workload set is available at the current stage. And it is also convenient to customize it by modifying the coordinator script of the corresponding workload set. One is to run the specified Workloads simultaneously, and the other is to continue to sequentially boot the instance holding a particular workload proportionally to a specific ratio every several seconds during the execution period. The combination of Workloads in a Workload set is firstly defined by an XML formatted configuration in advance. Every Workload is assigned with a digit number indicating the possibility of launching this workload at every provision stage. These numbers also play an important role when CloudScore calculates the final score for this run. This file is configurable to the tester, and it leverages the optimization aspects of the cloud system.

Parallel Mode The parallel mode will boot all instances at first with all necessary IP address or volume resources. Then it starts all the workloads at the same time and runs for a certain period. This gives a constant and diverse pressure on various components of the cloud system, and imitates the mixed behavior of different user applications.

Sequence Mode The sequence mode keeps booting new instances and runs the workloads at a certain rate, which gives an incremental load to the cloud in the beginning and may keep a dynamic balanced period if the whole running period is long enough. The number of instances running a certain workload

out of all instances is expected to be close to the provision ratio defined in the configuration file. This emulates more realistic usage scenarios with carefully controlled parameters.

Finally, CloudScore retrieves all the result packages via the master instance and calls the metric process method to analyze them. These four stages can be easily redefined by the tester.

Besides, new stages can be added and the default stages can be removed.

3.4 CloudScore Perform Unit

There are a lot of other important metrics presenting the overall quality of a cloud system, for example, the provision ability, the scalability, the reliability and so on. CloudScore defines CloudScore Performance Unit to simply the complex data calculation process. CloudScore Workload defines the necessary resources and the Coordinator Daemon is responsible for allocating and preparing them. Coordinator Daemon first checks whether all instances are accessible and if they are, it will ask the cloud to prepare the other necessary resources, such as public IP addresses and volumes, to continue the test process. Workload also specifies the necessary stages to complete the run, and by default there are four stages, Prepare, Run, Stop and Post. If the default workload flow is used, the Coordinator will start to call all the Prepare Stage to execute to deploy the workload under the proper environment automatically once the corresponding instances are prepared.

A CloudScore Performance Unit can be a single instance or a group of several instances holding the specified Workloads. Besides, CloudScore introduces a parameter named as Parallelism, which specifies the parallel degree of the Performance Unit.

4 Model

4.1 Performance Scale Analysis

A lot of different kinds of experiments are performed in different clouds including CSPU Parallel and Sequence Mode, with a lot of conditions and parameters changing. To aggregate the results and do performance comparisons of the clouds, a new method is introduced to illustrate the performance of cloud with the scale of workloads. In this model, the work scale changes with performance and it is divided into two segments. The turning point of the piecewise function is represented by d. The first half of the model indicates the relationship between the scale and performance with the shortage of resources. It is an exponential function, e suggests the changing tend, and a_1 can adjust the exponential relationship to describe the performance trend accurately. The second part of the model is the relationship between the scale and performance with enough resources. It is a polynomial function, adjusting the polynomial coefficient a_2, c, b to adjust the changing trend. This performance model is based on a series of experiments, and the experiments contains basic mode and complex mode. The basic mode focuses on one aspect such as computing power, network capacity, and it generally uses only one workload. Complex mode including static parallel mode or dynamic incremental mode uses a variety of workloads for a multifaceted evaluation of cloud system.

A scatter diagram is drawn with the performance value of different parameters in the x-axis and the level of parallel in the y-axis. Pairs of the performance metric(P) and the parallel level($Scale$) of the workload are composed. An interval of performance value is defined to divide the range of performance into segments. The pair representing the largest parallel level in the corresponding ranges of performance is picked out and every series of pairs are calculated to a trend line. For the multi-workloads case, a final metric, for example, the geometric mean of all the average value of the independent workloads, is defined as an aggregation metric for the whole system. The aggregation method is easy to customize in the coordinator script of the workload.

The pairs of performance value and parallel level present the performance levels in different test scenarios, and the overall trends of the scale are available for comparing and analyzing the capability and bottleneck of the clouds.

4.2 Cost Performance Analysis

A cost performance model is built based on the scale performance model mentioned above. The model describes the relationship of the cost of allocating some resources and the performance achieved with these resources. Basically, the cost is defined in different

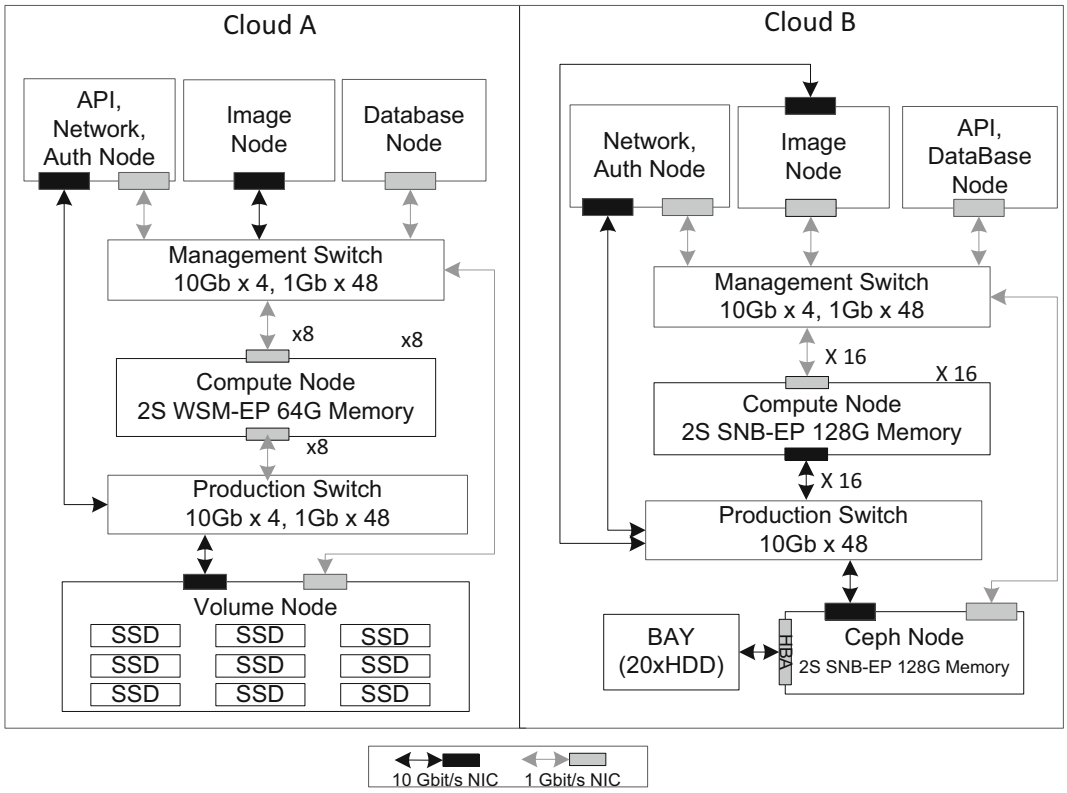

Fig. 2 Physical Architecture of Cloud A and B

ways in different clouds. The cost performance models can be used to distinguish the economic efficiency of the clouds.

The total cost of each experiment is calculated by adding the cost of every kind of resource after multiplying the execution time and the price per hour of one kind of resources. The performance is described in the same way with scale performance model. The data point of the pair of the cost and performance per workload instance is illustrated. Its trend line is fitted and summarized into the following formula.

5 Evaluation

Amazon EC2 is chosen as a representative of the public cloud, and two different OpenStack private clouds are built. A series of experiments are performed with CloudScore to evaluate and compare their computing capacity, network and storage ability, as well as provision ability and scalability. The architecture of the two private clouds as Cloud A and Cloud B is illustrated in Fig. 2 and details is shown in Table 2.

Cloud A has eight homogeneous compute nodes with two Intel®®Xeon® X5670 Westmere 2.93GHz processors. The machines in Cloud B are installed

Table 2 Physical environment

Config	Cloud A Core #, MemSize	Cloud B Core #, MemSize
Compute Node	12, 64GB	16, 128GB
Image Node	16, 32GB	16, 128GB
Volume Node	12, 64GB	16, 128GB
Database Node	12, 32GB	16, 128GB
Network Node	12, 16GB	16, 128GB
Compute Node #	8	16
Internal Network	1Gb/s	10Gb/s
Storage Back end	ISCSI	Ceph
Storage Capacity	576GB	18TB
OpenStack Version	Grizzly	Havana

Table 3 Openstack standard instance type

Types	vCPUs	RAM	Types	vCPUs	RAM
Small	1	2GB	Large	4	8GB
Medium	2	4GB	Xlarge	8	15GB

with two Intel® Xeon® E5-2670 Sandy Bridge-EP 2.6GHz processors. Extreme x670V Switch is used in Cloud B to provide enough 10Gbit/s links for the compute nodes and others are Helion ProCurve 2910al-48G-PoE+ J9148A Switches. All servers are hyper-threading enabled. Cloud A uses iSCSI for storage back end, while Cloud B uses Ceph 0.72.2. All machines in Cloud A are installed with Ubuntu 12.04.3 LTS and CentOS release 6.4 in Cloud B. The images for booting instances are based on Ubuntu in Cloud A and CentOS in Cloud B. The OpenStack architectures are deployed with default parameters and it is scheduled to spread instances evenly on all the compute nodes to ensure load-balancing.

5.1 Hadoop on Amazon

Several experiments are performed on the Amazon EC2 platform, by applying the Hadoop benchmarks, wordcount and kmeans from HiBench. The Hadoop cluster is defined with one Hadoop controller node and several Hadoop slave nodes. The Hadoop controller runs as the Name Node and Resource Manager, while the Hadoop slave nodes run as the Data Nodes and Node Managers. The instance types used in these experiments are summarized in Table 3.

Each workload with each instance type are run for three types. The Hadoop cluster has 51 nodes. One node is used as a name node of the distributed file system level and a resource manager of application level.

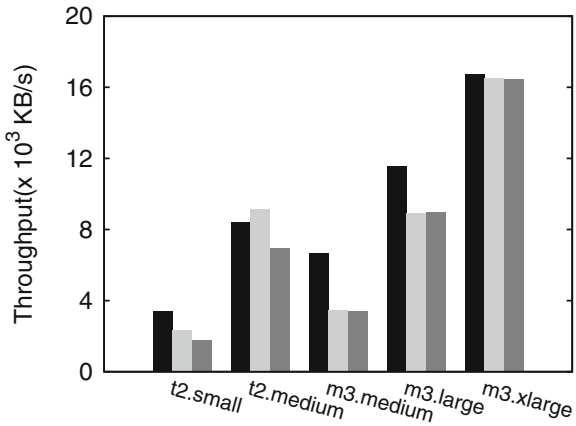

Fig. 3 Throughput of kmeans

It is also added in calculation and data processing as a data node and a node manager. But the other 50 nodes are only used as data nodes and node managers. The results of kmeans running in Amazon EC2 service are shown in Table 4.

The datasize is set to 32G and the throughput, time for data prepare, time for running and time for provision are shown respectively in Figs. 3, 4, 5 and 6. The result of wordcount are not shown in this section, but it is used to build the cost performance model.

Figure 3 shows that the throughput seems in proportion to the vCPUs numbers of the instance type. It is because that the kmeans is CPU intensive. Also Figs. 4 and 5 indicate more vCPU and more memory reduce the preparation and run time for the workload. The repeated experiments with the same instance type all show some fluctuation in the results of throughput, prepare time and run time, and larger the instances are, smaller the fluctuation is. The provision time results also show some instability, while it seems less concerned about the instance type.

Table 4 Results of kmeans run in amazon ec2 service

Instance Config	KMEANS				WordCount	
Type	vCPUs	RAM GB	Perf MB/s	RunTime second	Perf MB/s	RunTime second
t2.small	1	2	3.3	2413	10.8	1875
t2.medium	2	4	8.9	894	66.5	825
m3.medium	1	3.75	3.3	2347	29.3	1325
m3.large	2	7.5	8.8	900	76.5	802
m3.xlarge	4	15	16.1	490	137.1	434

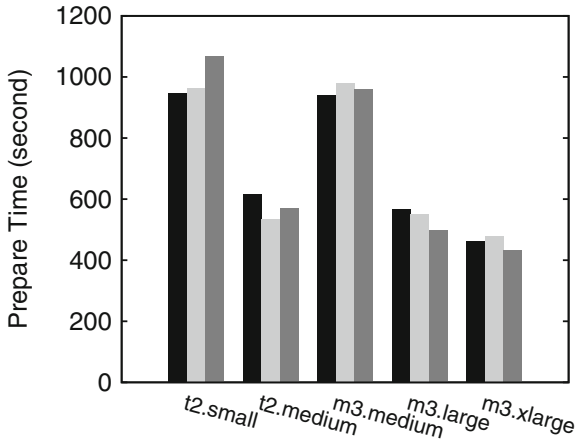

Fig. 4 Prepare Time of kmeans

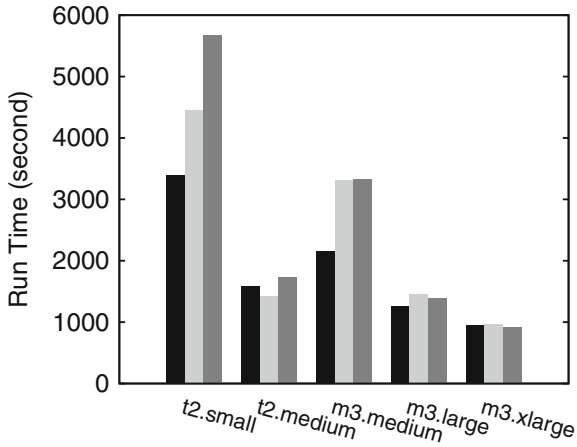

Fig. 5 Run Time of kmeans

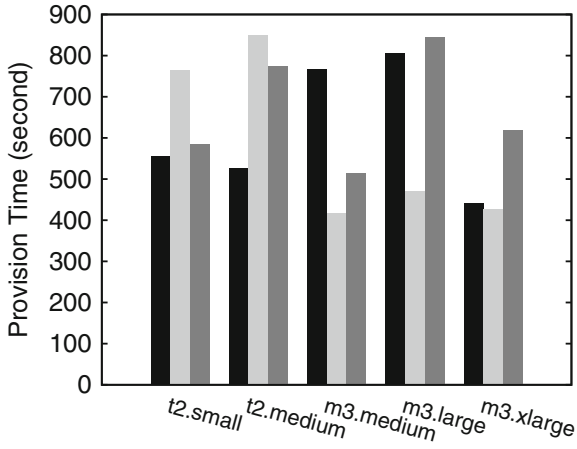

Fig. 6 Provision Time of kmeans

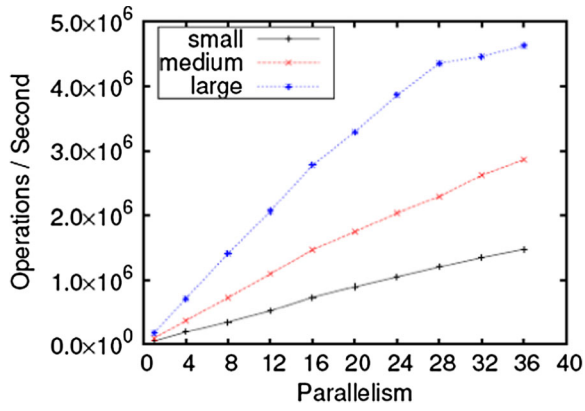

Fig. 7 Throughput of Java in Parallel

5.2 Parallel Mode

CloudScore provides more complex test scenarios involving several workloads which gives a more comprehensive view of the cloud. The Parallel Mode boots and prepares the instances and resources in advance, and starts to run different workloads at the same time. Java, WebServer and Database workloads are grouped as a Parallel workload in this section. It is run on different instances at different parallel levels in Cloud B. It is also defined two different availability zones. For the WebServer workload, the server instances and client instances are provisioned in the different availability zones. The other two workload boot their instances in both the zones. Thus, the total number of instances provisioned in the cloud actually is four times that of the parallelism shown in Figs. 7, 8 and 9.

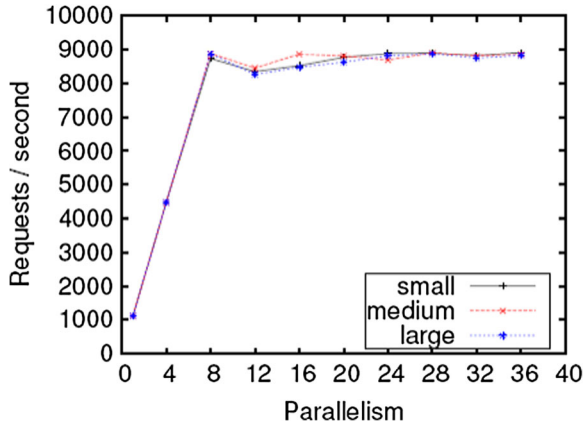

Fig. 8 Throughput of WebServer in Parallel

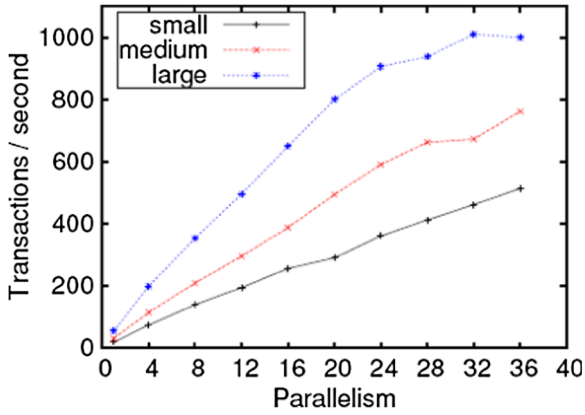

Fig. 9 Throughput of Database in Parallel

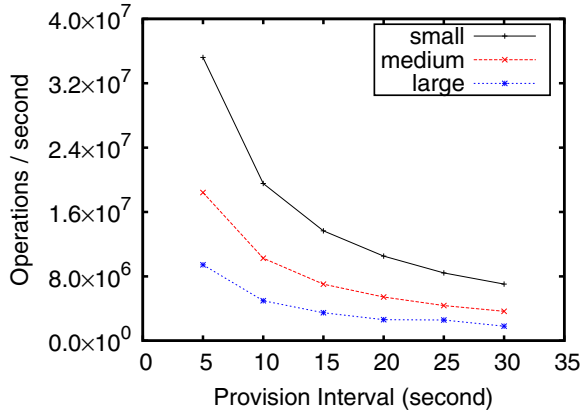

Fig. 10 Throughput of Java in Sequence

When the parallelism reaches 32, there are actually 128 instances launched in the cloud (32 for Java workload), and each compute node hosts 8 instances because of the load balancing scheduling methodology. When running with large instances (4 vCPUs), each physical core in the host is shared by 4*8/16=2 vCPUs. But the throughput (Operations per second) does not decrease drastically, because the other two workloads are not CPU-intensive.

The throughput of Web workload reaches the peak of all instance types at parallel level 8. Because the cloud does not limit the available bandwidth for the instance types and due to the best effort network principle, all bandwidth are occupied at parallel level 8. When parallel level reaches 8, the instances running Web workload have to share the physical link with co-host instances, which leads to the drop of throughput between each client-server pair. In spite of the fact that the total throughput of the Java workload still grows slightly after parallel level 28, the aggregate throughput of Database workload drops at parallel level 32, which means the bandwidth of remote volumes becomes the bottleneck.

5.3 Sequence Mode

The Sequence Mode runs a set of workloads in a simultaneous way by provisioning workloads at a defined speed, say 5 seconds or 15 seconds. The workload may continue to run for several minutes, so the cloud will keep dynamically stable after some ramp up period. The set of workloads in this section also contains the Java, WebServer and Database sub-workloads. All the running stages of the sub-workloads last for 300s. During the steady period of the run, the shorter the provision interval is, the more the number of instances alive in the cloud varies. When the interval is 5s, the number of instances ranges from 80 to 100, while the number of instances keeps near 30 at 15s-interval. And in all experiments, it takes about 400s for the cloud to get into the steady state, when the first boot workload finishes its execution.

All the experiments set the provision ratio *Java:WebServer:Database* to 4:3:3, and the workload is decided randomly according to the ratio at every provision time. The actual ratio does not apply perfectly to 4:3:3 due to the randomness of decision and a quite small provision ratio of WebServer with the interval as 25s on small instances is observed. This is because the provision interval is not small enough to provision more workloads. The actual ratio gets closer to the theoretical ratio with smaller provision interval such as 5s. The aggregated results of the throughput of three workloads on three kinds of instances are shown in Figs. 10, 11 and 12. With provision interval becomes shorter, the average performance of all the three workloads drops. But the decrease of performance is not very outstanding, except for WebServer, which leads to the rise of the aggregation results of shorter provision interval experiments.

5.4 Scale Performance Model

The method mentioned in Section 3 is used to evaluate the scalability of Cloud A and B. Figure 16

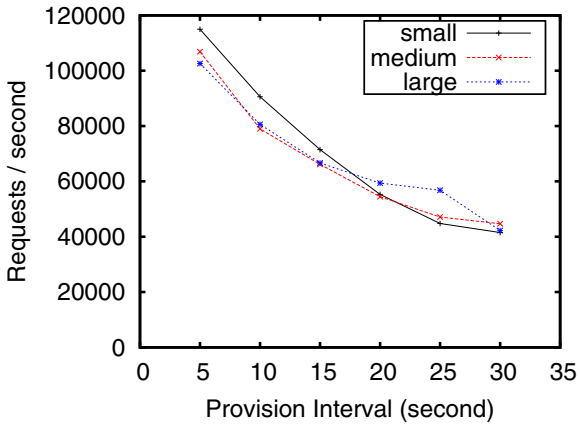

Fig. 11 Throughput of WebServer in Sequence

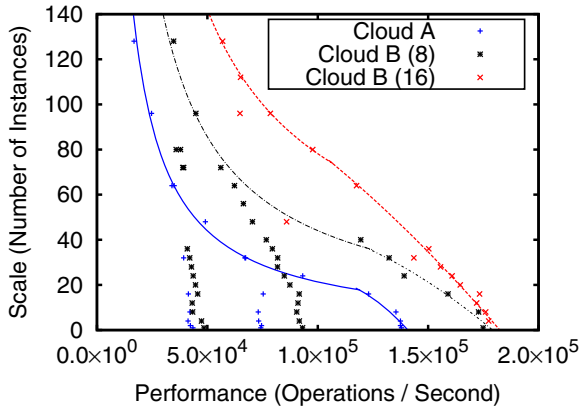

Fig. 13 Model based on Java

shows the aggregation of the Parallel tests on medium instances in Cloud A, CSPU Parallel and Sequence tests on small, medium and large instances in Cloud B applying. The aggregation method is defined as the geometric mean value of the average throughput of sub-workloads. Cloud A shows poor overall performance because of its smaller network bandwidth (1Gbit/s), and poor scalability because that the storage networking collapses quickly when parallel level increases. Cloud B Sequence and Parallel tests show the similar tendency, as well as better performance and scalability.

To analyze different aspects of the cloud ability, the results of Java, WebServer and Database workload are shown respectively in Figs. 13, 14 and 15.

There are three groups of points in Fig. 13, representing the average operation throughput in one test

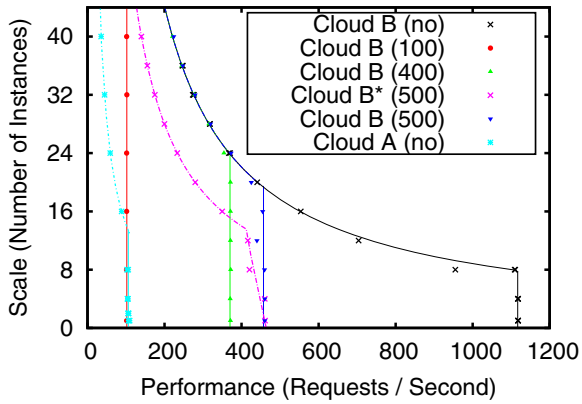

Fig. 14 Model on WebServer

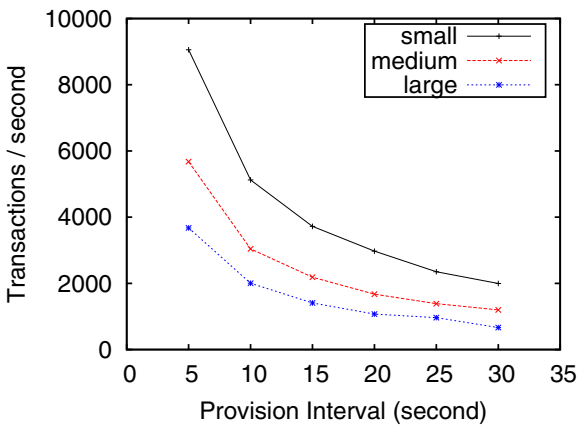

Fig. 12 Throughput of Database in Sequence

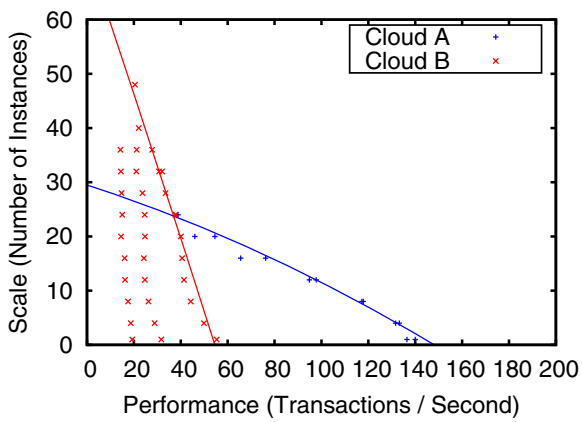

Fig. 15 Model based on Database

in Cloud A and in Cloud B with 8 compute node enabled and 16 compute node enabled. Every group indicates that with the increase of parallelism, the average performance of large instances gets closer to that of medium instances, and so is medium to small instances.

Figure 14 shows the average throughput of WebServer workload with assigning different bandwidth quota to the instances in Cloud A and Cloud B. If there is no limit to the bandwidth of the instances, Cloud A and Cloud B will occupy all the bandwidth of the physical link, respectively 1Git/s and 10Git/s at the low scale. If the physical link is shared, the average throughput becomes in inverse proportional to the parallel level, because the total bandwidth is fixed. The inverse coefficient is proportional to the sum of bandwidth of all compute nodes. With bandwidth quota assigned to the instances, the Cloud B provides good isolation among the co-hosted instances, and the average throughput does not collapse until some link is shared.

$$S = \begin{cases} 3.50 \times 10^6 P^{-1.04} & P < 1.20 \times 10^5 \\ -1.25 \times 10^{-8} P^2 + 2.24 \times 10^{-3} P - 95.11 & P \geq 1.20 \times 10^5 \end{cases} \quad (1)$$

$$S = \begin{cases} 2.75 \times 10^6 P^{-0.96} & P < 1.26 \times 10^5 \\ -3.24 \times 10^{-9} P^2 + 3.39 \times 10^{-4} P + 43.28 & P \geq 1.26 \times 10^5 \end{cases} \quad (2)$$

$$S = \begin{cases} 1.57 \times 10^6 P^{-0.86} & P < 1.09 \times 10^5 \\ -2.11 \times 10^{-9} P^2 + 3.65 \times 10^{-4} P + 136.76 & P \geq 1.09 \times 10^5 \end{cases} \quad (3)$$

Database Workload is also performed on medium instances in Cloud A and three kinds of instances in Cloud B. Cloud A provides much better performance when the parallel level is low, but the average throughput collapses rapidly in large scale in Fig. 15. In contrast, Cloud B provides a relatively stable storage ability and the throughput does not decrease very much in large scale. This is the difference caused by the different back ends of their remote storage solution. Figure 16 shows the relationship between the overall performance and the number of virtual machines in Cloud A and Cloud B experiments.

5.5 Cost-Performance Modeling

The price method of Amazon EC2 is chosen to evaluate the cost performance model introduced in Section 4. The model is built by calculating all costs of all instance types. Figures 17, 18 and 19 show the models built on individual workload experiments of Java Server, Web Server and OLTP Database respectively.

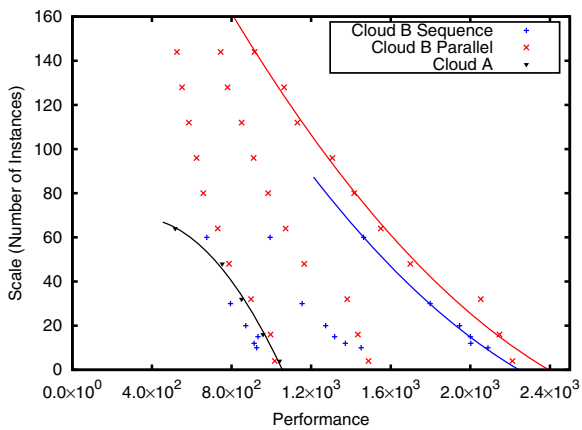

Fig. 16 Geometric mean to model

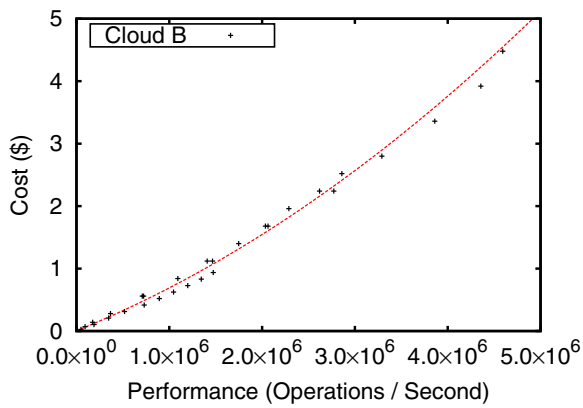

Fig. 17 Cost model based on Java

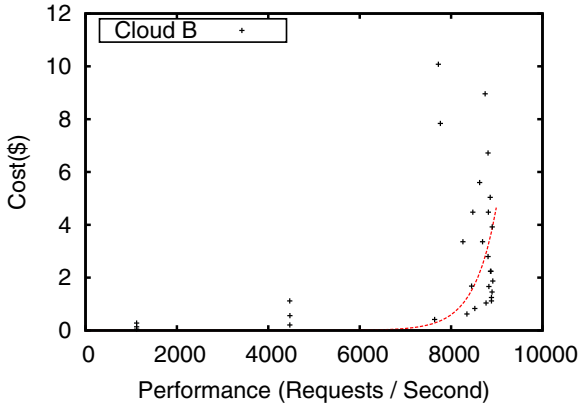

Fig. 18 Cost model based on WebServer

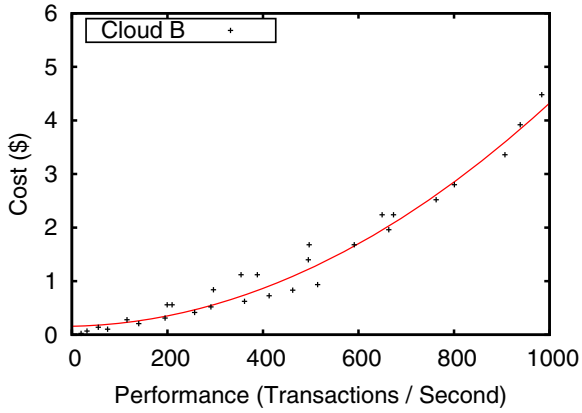

Fig. 20 Cost model based on geometric

Figure 20 gives the overall cost-performance relation, with getting the total cost of every individual instances and getting the overall performance by calculating the geometric mean value of the average performance of every workload. The cost-performance relations based on Java Server, OLTP Database and the overall case are illustrated in Formula 7, 8 and 9.

$$Cost = 8.44\times10^{-14}P^2+6.01\times10^{-7}\times P+0.01 \quad (4)$$

$$Cost = 4\times10^{-6}\times P^2+1.65\times10^{-4}\times P+0.16 \quad (5)$$

$$Cost = 1.97\times10^{-8}\times P^2-1.62\times10^{-4}\times P+0.93 \quad (6)$$

The cost seems in proportion to the performance in Java Server model, but the costs rise rapidly with larger performance in OLTP Database model and overall model.

Besides, the cost-performance model is also built on the Hadoop workloads. Figures 21 and 22 illustrate the cost-performance model by applying workdcount and kmeans to t2 and m3 family of instances in Amazon EC2. Formula 10 and Formula 11 shows the formula of the trendlines for instance type class t2 and m3 of wordcount, and Formula 12 and Formula 13 shows that of kmeans of t2 and m3 family.

$$Cost = 1.78\times10^{-6}\times P^2-3.74\times10^{-4}\times P+0.04 \quad (7)$$

$$Cost = 3.84\times10^{-6}\times P^2-3.80\times10^{-4}\times P+0.06 \quad (8)$$

The m3 family costs more for the same performance, because Amazon increases its price for more

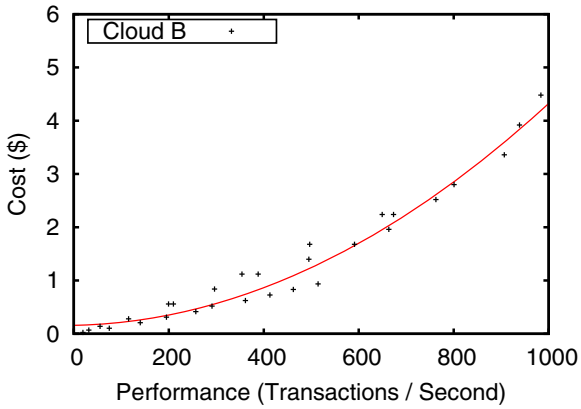

Fig. 19 Cost model based on Database

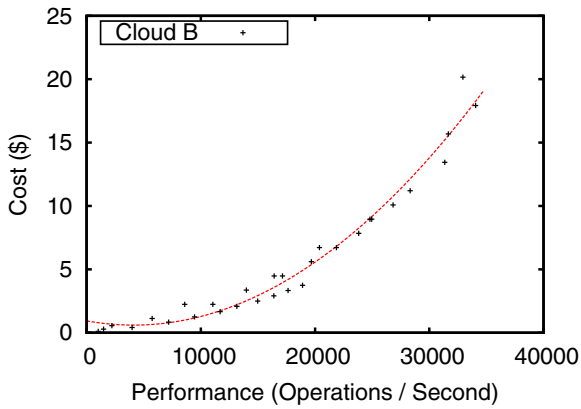

Fig. 21 Cost model based on wordcount

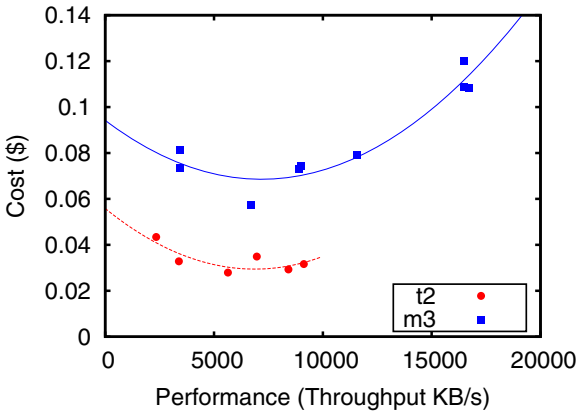

Fig. 22 Cost model based on kmeans

disk resources, which neither wordcount nor kmeans asks for.

$$Cost = 5.59\times10^{-10}\times P^2 - 7.69\times10^{-6}\times P + 0.06 \quad (9)$$

$$Cost = 4.98\times10^{-10}\times P^2 - 7.15\times10^{-6}\times P + 0.09 \quad (10)$$

6 Conclusion

A benchmark CloudScore is proposed and it targets at assessing the cost performance of IaaS clouds. CloudScore aims at automatically cloud resource management for multiple clouds and flexible workload extension. Based on the open source, the cloud API module are realized that the parameters can be automatically collected and calculated. A series of experiments are also carried out to show how it works to identify the problems of a small scale private cloud. The convenience and efficiency of CloudScore benefit testers to carry out their experiments by comparing cost performance values on different clouds.

Acknowledgments This work was supported in part by National Nature Science Foundation of China (61073151, 61572195), Shanghai Agriculture Science Program (2015.3-2), and Shanghai Key Laboratory of Trustworthy Computing Open Project Fund (07dz22304201608).

References

1. Bhaskar Prasad, R., Admela, J., Dimitrios, K., Yves, G.: Architectural Requirements for Cloud Computing Systems: An Enterprise Cloud Approach. J. Grid Comput. **9**(1), 3–26 (2011)
2. Thomas, R., Geoff, C., et al.: Grid and cloud computing: opportunities for integration with the next generation network. J. Grid Comput. **7**(3), 375–393 (2009)
3. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010)
4. Mell, P., Grance, T.: The nist definition of cloud computing. NIST Spec. Publ. **800**(145), 7 (2011)
5. Wang, L., Von Laszewski, G., Younge, A., He, X., Kunze, M., Tao, J., Fu, C.: Cloud computing: a perspective study. New Gener. Comput. **28**(2), 137–146 (2010)
6. Costa, P., Migliavacca, M., Pietzuch, P., Wolf, A.L.: Naas: network-as-a-service in the cloud. In: Proceedings of the 2nd USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, Hot-ICE, vol. 12, pp. 1-1. USENIX, Berkeley (2012)
7. Shiraz, M., Gani, A., Azra, S., Khan, S., Ahmad, R.W.: Energy efficient computational offloading framework for mobile cloud computing. J. Grid Comput. **13**(1), 1–18 (2015)
8. Yangui, S., Marshall, I.J., Laisne, J.P., Tata, S.: CompatibleOne: the open source cloud broker. J. Grid Comput. **12**(1), 93–109 (2014)
9. van Vliet, J., Paganelli, F.: Programming Amazon EC2. O'Reilly Media, Sebastopol (2011)
10. Krishnan, S.: Programming Microsoft Azure. O'Reilly Media, Sebastopol (2010)
11. Hp helion public cloud. http://www.hpcloud.com/
12. Sabharwal, N., Shankar, R.: Apache CloudStack Cloud Computing. Packt Publishing, Birmingham (2013)
13. Toraldo, G.: OpenNebula3 Cloud Computing. Packt Publishing, Birmingham (2012)
14. Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D.: The eucalyptus open-source cloud-computing system. In: 2009 9th IEEE International Symposium on Cluster Computing and the Grid, CCGRID'09, IEEE, pp. 124–131. IEEE, Piscataway (2009)
15. Jackson, K., Bunch, C.: OpenStack Cloud Computing Cookbook, 2nd edn. Packt Publishing, Birmingham (2012)
16. Avetisyan, A.I., Campbell, R., Gupta, I., Heath, M.T., Ko, S.Y., Ganger, G.R., Kozuch, M.A., O'Hallaron, D., Kunze, M., Kwan, T.T., et al.: Open cirrus: a global cloud computing testbed. Computer **43**(4), 35–43 (2010)
17. Keqin, L.: Optimal load distribution for multiple heterogeneous blade servers in a cloud computing environment. J. Grid Comput. **11**(1), 27–46 (2013)
18. Ge, X., Qi, Z., Chen, K., Duan, J., Dong, Z.: Loosely-coupled benchmark framework automates performance modeling on iaas clouds. In: Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, UCC 2014. IEEE, Piscataway (2014)
19. Casazza, J.P., Greenfield, M., Shi, K.: Redefining server performance characterization for virtualization benchmarking. Intel Technol. J. **3**, 10 (2006)
20. Makhija, V., Herndon, B., Smith, P., Roderick, L., Zamost, E., Anderson, J.: Vmmark: a scalable benchmark for virtualized systems. VMware Inc, CA, Tech. Rep. VMware-TR-2006-002 (2006)

21. Muñoz, V.M., Ramo, A.C., Diaz, R.G., Tsaregorodtsev, A.: Cloud governance by a credit model with DIRAC. In: Proceedings of the 4th International Conference on Cloud Computing and Services Science, pp. 679–686 (2014)
22. Zheng, X., Martin, P., Brohman, K., Xu, L.D.: Cloud service negotiation in internet of things environment: A mixed approach. IEEE Trans. Ind. Inf. **10**(2), 1506–1515 (2014)
23. Tao, F., Laili, Y., Xu, L., Zhang, L.: Fc-paco-rm: A parallel method for service composition optimal-selection in cloud manufacturing system. IEEE Trans. Ind. Inf. **9**(4), 2023–2033 (2013)
24. Wang, L., Zhan, J., Luo, C., Zhu, Y., Yang, Q., He, Y., Gao, W., Jia, Z., Shi, Y., Zhang, S., Zheng, C., Lu, G., Zhan, K., Li, X., Qiu, B.: Bigdatabench: A big data benchmark suite from internet services. In: Proceedings of the 20th IEEE International Symposium on High Performance Computer Architecture, HPCA'2014. Piscataway, NJ, USA: IEEE, pp. 488–499 (2014)
25. Zheng, X., Martin, P., Brohman, K., Xu, L.D.: Cloudqual: A quality model for cloud services. IEEE Trans. Ind. Inf. **10**(2), 1527–1536 (2014)
26. Tao, F., Zuo, Y., Xu, L.D., Zhang, L.: Iot-based intelligent perception and access of manufacturing resource toward cloud manufacturing. IEEE Trans. Ind. Inf. **10**(2), 1547–1557 (2014)
27. Muñoz, V.M., Ramo, A.C., Albor, V.F., Diaz, R.G., Arévalo, G.M.: Rafhyc: an architecture for constructing resilient services on federated hybrid clouds. J. Grid Comput. **11**(4), 753–770 (2013)
28. Cloudharmony: simplify the comparison of cloud services, Laguna Beach, CA, USA, http://cloudharmony.com/
29. Li, A., Yang, X., Kandula, S., Zhang, M.: Cloudcmp: comparingpublic cloud providers. In: Proceedings of the 2010 conferenceon Internet measurement conference, ACM. NewYork, NY, USA: ACM, pp. 1–14 (2010)
30. Novakovic, D., Vasic, N., Novakovic, S., Kostic, D., Bianchini, R.: Deepdive: Transparently identifying and managing performance interference in virtualized environments. In: Proceedings of USENIX ATC'13: 2013 USENIX Annual Technical Conference, ATC'13. Berkeley, CA, USA: USENIX, pp. 219–230 (2013)
31. Ferdman, M., Adileh, A., Kocberber, O., Volos, S., Alisafaee, M., Jevdjic, D., Kaynak, C., Popescu, A.D., Ailamaki, A., Falsafi, B.: Clearing the clouds: a study of emerging scale-out workloads on modern hardware. In: ACM SIGARCH Computer Architecture News, vol. 40, no. 1, ACM. NewYork, NY, USA: ACM, pp. 37–48 (2012)
32. El-Refaey, M.A., Rizkaa, M.A.: Cloudgauge: a dynamic cloud and virtualization benchmarking suite. In: 2010 19th IEEE International Workshop on Enabling Technologies: Infrastructures for Collaborative Enterprises, IEEE. Piscataway, NJ, USA: IEEE, pp. 66–75 (2010)
33. Silva, M., Hines, M.R., Gallo, D., Liu, Q., Ryu, K.D., da Silva, D.: Cloudbench: Experiment automation for cloud environments. In: Proceedings of the 2013 IEEE International Conference on Cloud Engineering, IC2E'13, IEEE. Piscataway, NJ, USA: IEEE, pp. 302–311 (2013)
34. Openstack Rally project, https://wiki.openstack.org/wiki/Rally
35. Fuel: Openstack deployment and management," Mountain View, CA, USA, http://docs.mirantis.com/openstack/fuel/fuel-5.1/planning-guide.html