Mobile Netw Appl (2017) 22:153–160
DOI 10.1007/s11036-016-0738-0

Cost-Effective Service Provisioning for Hybrid Cloud Applications

Fangming Liu[1] · **Bin Luo**[1] · **Yipei Niu**[1]

Published online: 17 May 2016

Abstract The workloads of social computing are tremendous, bursty and unpredictable. A hybrid cloud, which combines a private cloud and a public cloud, is a promising solution to processing the workloads. For most corporations, they leverage one public cloud. However, with fierce competition among public cloud providers, public cloud services change frequently, which may lead to service unavailability and a less cost-effective hybrid cloud solution. As a result, leveraging multiple public clouds in the hybrid cloud is a potential solution. In this paper, we identify such a problem in current hybrid cloud and analyze the necessity of load balancing for hybrid cloud applications. Focusing on cost minimization and performance guarantee, we propose a Least Cost per Connection (LCC) algorithm so as to choose the most cost-effective clouds along with adapting changes among multiple public clouds. The simulation results show that our solution can significantly decrease the outsourcing cost as well as guarantee Quality of Service (QoS) of applications.

Keywords Hybrid cloud · Multi-cloud · Load balancing · Cost-effective

✉ Fangming Liu
fmliu@hust.edu.cn

[1] Key Laboratory of Services Computing Technology and System, School of Computer Science and Technology, Huazhong University of Science and Technology, Hubei, China

1 Introduction

Social computing, which supports social network services, such as YouTube, Facebook, Whatsapp and so on, has become increasingly important. Dealing with the workloads generated by social networking services is challenging. For example, news feed in Facebook, which updates the news from people that you follow, includes various types of contents. On the one hand, huge amounts of users are using Facebook and interacting with each other simultaneously, which generates tremendous workloads. On the other hand, the workloads are closely related to users' daily lives. Hence, the workloads vary with time, which may make the private cloud of an IT corporation overload at noon yet idle at midnight.

The private cloud, which is supported by dedicated datacenters or server clusters, can provision enhanced security and ultimate control. However, the private cloud has a limited capacity and low scalability. When workloads increase, it is difficult to address the overload problem with the private cloud. As the public cloud can provision elastic computing resources. It can help the private cloud process the excessive workloads flexibly with low costs. Hence, a hybrid cloud, which combines a private cloud and a public cloud, is a promising solution to processing the bursty and tremendous workloads generated by social applications. The state of the cloud report from RightScale points out that, among all the enterprise respondents, 55 % of them expect to use hybrid clouds [5].

However, for most corporations, they deploy their applications on single public cloud. First, cloud computing platforms sometime breakdown and update bug, which may influence their business. Second, cloud providers offer

discount to attract users and occupy cloud computing market. For example, the price of AWS drops 12 times in 2013 while Google Compute Engine [2] made a cumulative reduction of 38 percent in prices from January to October in 2014. When the price drops, deploying applications in single cloud makes it difficult to switch to other providers for lower price. Third, since more corporations join in the cloud market, cloud users are willing to avoid provider lock-in. As a result, deploying applications in multiple public clouds can contribute to obtaining a more cost-effective and stable hybrid cloud solution.

When leveraging multiple public clouds, we face the following challenges. First, since cloud users can leverage multiple clouds, it is challenging to distribute workloads among private and public clouds so as to obtain a cost-effective solution. Second, as cloud computing platforms sometime breakdown or update, it may lead to service unavailability or performance degradation. Such maintenance makes it more difficult to derive a cost-effective solution. Third, cloud computing price changes with time. Cloud users always want to use the most cost-effective cloud products. Deploying applications on unsuitable cloud products may cost more money and make applications inefficient.

To address these problems, we propose a cost-effective service for hybrid cloud applications, which selects the best public cloud for outsourcing and adapts cloud price changes dynamically, along with provisioning global load balancing. The system uses a two-tier load balancing mechanism, provisioning virtual machine (VM) and cloud level load balancing. The load balancing problem [8] for hybrid cloud applications is different from the cloud selection problem. Heterogeneous workload characteristics as well as divergent cloud instance types complicate the problem of identifying the best suitable cloud provider and instance for a specific application. Furthermore, a cloud user may have several applications and every application has many choices on cloud selection, which makes the scheduling more complex. The system firstly chooses the best instance on each cloud for certain applications using CloudCmp [6]. Then the system uses the proposed Least Connection per Cost (LCC) algorithm to distribute job requests among public clouds. At last, the system scales up or down automatically according to the price and performance of each cloud.

In this paper, we proposed a cost-effective service provisioning system for hybrid cloud applications, which provides functions as follows:

1. guaranteeing the private cloud resource use first;
2. selecting the best public cloud for out-sourcing according to the request type;
3. adapting dynamically cloud price changes;
4. adding or removing cloud provider at will;
5. provisioning global load balancing.

The remainder of this paper is organized as follows. In Section 2, we introduce the system architecture and design objectives. Section 3 provides detailed description of implementation principles and proposes LCC job scheduling algorithm. Section 4 introduces a series of simulation experiments to test our system functions and performance. Section 5 provides an overview of related work. Finally, we conclude our paper and provide future works in Section 6.

2 Design objectives

This system is designed to help the hybrid cloud users provision cost-effective services, especially for those who deploy multiple public clouds. Based on the fact that different types of applications require different computing resources, we classify all the applications into three representative types roughly:

* *CPU-intensive application*. In these applications, CPU resource is more eagerly needed than other computing resource. For example video processing, scientific computing and so on.
* *Memory-intensive application*. High IO throughput application such as large Map-Reduce tasks which depend on sufficient memory for data shuffling.
* *Disk-intensive application*. NoSQL database (e.g., Cassandra, MongoDB) or Distributed File System (e.g., HDFS) have high demand on storage.

Our design prefers processing workloads in the private cloud, and outsourcing excessive workloads to the public clouds. Since prices and performance of public clouds vary from cloud providers, we need to maintain high quality services as well as save cost. Before the load balance system start working, we select the most suitable type of instances on each cloud in advance for the application. Since Li *et al.* [6] compare VMs of multiple cloud providers so as to find out which are the most suitable.

Figure 1 shows an overview of load balance system design for hybrid cloud applications. First, cloud users need to register their applications on the load balancer with a unique application id and provide detailed information of requested resource. Second, through administrating an application-cloud mapping table, the system selectively adds a private cloud and public clouds for certain applications.We deploy the application with cloud agent and generate a virtual machine template on the most suitable

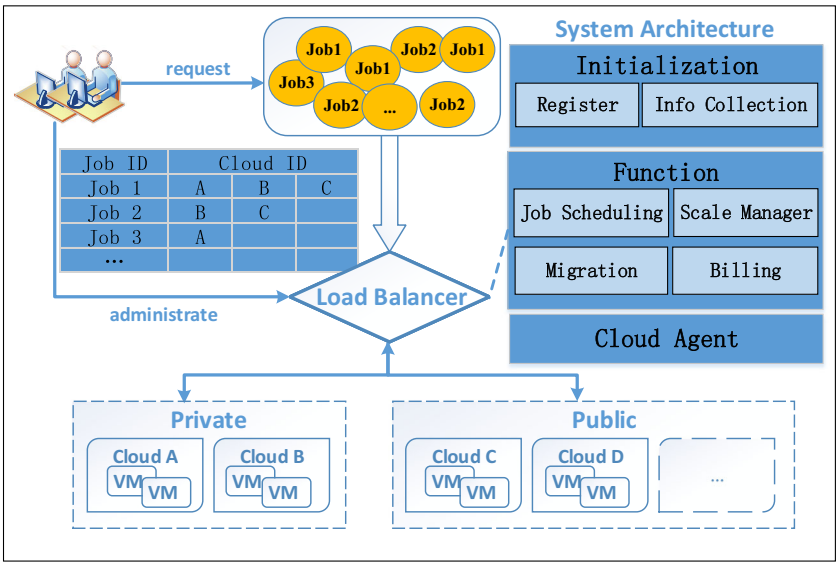

Fig. 1 Overview of load balance system design for a hybrid cloud

instance type. Cloud agent starts to collect and feedback the cloud load information to load balancer periodically.

The load balancer receives feedbacks and updates the priority of each cloud based on our load balance algorithm. Here the cycle length depends on the job request arrival rate. When the load balancer receives a batch of job requests, it checks the priority of each cloud and dispatches the job requests to the most cost-effective clouds.

Cost-effective Due to the unpredictable workloads, it's hard for a private cloud to predict an exact amount of hardware resources. As such, the private cloud is not capable of processing such immense workloads, and the public clouds can satisfy the requirements of high capacity and scalability. Considering the variety of cloud providers, users want to add or remove a cloud for their applications freely. When users deploy their applications on multiple clouds, how to choose the most cost-effective clouds becomes a critical problem.

One reason stops users to use public cloud is the high cost to afford. Many enterprises would rather struggle for building their private cloud other than totally move their applications onto public cloud. However, it's hard for a private cloud procure a known amount of hardware and software. Even more, there will inevitably be IT accidents or flash crowds like The Shopping Day, private cloud can't meet such unforeseen emergency. Hybrid cloud load balancer must provision cost-effective service to solve this problem. For the sake of cloud users, we desire to add or delete a cloud for a certain application at will when we don't think it cost-effective. Therefore, which clouds are more cost-effective for the current application become a problem.

In this paper, how much a job request costs is taken as the measurement of cost-effectiveness (CE ratio for short). For example, if the price of a VM of a cloud provider is P, and the VM can process N job requests per second, the CE ratio equals $\frac{P}{N}$. The number N is hard to figure out through calculation, so we use measurement tools of Cloudcmp to monitor it.

Flexibility With rapid development of cloud computing market, services provisioned by public cloud providers change quite frequently. As such, cloud users need to change their strategies of leveraging public clouds.

In order to achieve this goal, the design of the system is guided by the asynchronous message-driven paradigm through RESTful design principle. We use RESTful API to reduce the interdependency of tightly coupled interfaces, generally lowering the complexity of integration.

Specifically, the system allows cloud users to choose on which clouds to deploy their applications, which leads to a many-to-many relationship.

Global load balance Users' applications are deployed on multiple clouds. As such, we need to ensure global load balancing, so as to provision cost-effective services and make the best use of cloud resources we've bought.

Since load information collected from other clouds is transmitted via the Internet, the transmit latency must be low. We use short TCP connections for immediate reinforcement. To reduce the number of messages, we abstract a cloud as an unit to distribute workloads among clouds. For load balancing among VMs, each cloud can address it.

Table 1 Key parameters

Notation	Definition
$\lambda(t)$	Average arrival rate of requests during the tth time slot
$d(t)$	Average service rate during the tth time slot
$\eta_i(t)$	The value of priority associated with each cloud during tth time slot
u_m	mth cloud of public clouds deployed in the hybrid cloud
r_m	mth cloud of private clouds deployed in the hybrid cloud
σ_u	Upper threshold of scaling up
σ_d	Lower threshold of scaling down
$C_i(t)$	the current number of connections on each cloud i during tth time slot
$M_i(t)$	The job request service rate of each VM in a cloud during the tth time slot
$P_i(t)$	The price of selected instance on cloud i during the tth time slot

The low-level quality of service is guaranteed by the service level agreement of cloud providers (Table 1).

3 System model

In this section we specifically describe how this system scheduling jobs and leveraging the auto scaling service for resource reallocation.

3.1 Job scheduling & resource allocation decoupling

Job scheduling and resource allocation are the two main tasks of load balancing. Job scheduling is highly required on bandwidth delay, while resource allocation is closely related to cost. Based on the characteristics of two different tasks, we discrete the job scheduling module and resource management module from logic view. Specifically, the job scheduling module only takes responsibility of receiving and dispatching job requests while resource allocation module takes charge of monitoring the cloud resource usage and decides on when to scale up or down.

3.2 Cloud-level load balancing & VM-level load balancing decoupling

In our system, we adopt a two-level hierarchical load balancing architecture, i.e., a cloud level and a VM level. On the cloud level, we take the cloud as an unit of scheduling object as shown in Fig. 1. On the VM level, we use cloud back-end load balancing services for job distribution among VMs. Almost all the public cloud providers provision load balancing services within their clouds. For example, Openstack [4] integrates LBaaS (Load-Balancing-as-a-Service) into Neutron component. LBaaS allows cloud users to scale their applications, detect unhealthy VM instances, balance loads across regions, route traffic to the closest VM and so on.

3.3 Centralized management

Cloud agent is used to sent cloud information back to the load balancer. We add a heartbeat mechanism to report the cloud health status periodically.

3.4 Algorithm

3.4.1 Cloud capability measurement

Here we consider two sets of cloud resources, i.e., PUB and PRI. Let PUB be the set of public clouds, which is denoted as $\{u_1, u_2, ..., u_m\}$. Moreover, let PRI be the set of the private cloud, which is denoted as $\{r_1, r_2, ..., r_n\}$. Furthermore, we set the resource parameters{cpu, mem, disk, net}.

$$M_i(t) = \min\{\frac{\lambda_i^{CPU}(t)}{d_i^{CPU}}, \frac{\lambda_i^{MEM}(t)}{d_i^{MEM}}, \frac{\lambda_i^{DISK}(t)}{d_i^{DISK}}, \frac{\lambda_i^{NET}(t)}{d_i^{NET}}\} \quad (1)$$

Equation 1 figures out the job request service rate of each VM in a cloud.

3.4.2 Job scheduling

We denote the current number of connections on each cloud as $C_i(t)$. And the price of selected instance on each cloud is denoted by $P_i(t)$. Furthermore, the value of priority of each cloud is denoted as $\eta_i(t)$.

$$\eta_i(t) = \frac{M_i(t) - C_i(t)}{P_i(t)}. \quad (2)$$

We sort the priority obtained based on Eq. 2 of each cloud, and record the priority list in configuration files. When a job request arrives, the load balancer will select the cloud which has the maximum priority to server the request.

If the heartbeat packet from the preferred cloud does not arrive in time, the preferred cloud will be the second one. The detailed algorithm is described as Algorithm 1.

Algorithm 1 Least Connection-Cost Ratio Scheduling (LCC)

```
Denote (key,index) as cloud type and cloud ID
for each time slot t ∈ [0, 1, 2, ...] do
  key = 0
  for each cloud r_i ∈ PRI do
    if r_i is not alarmed then
      set key = 1
      calculate cloud priority η_i
    end if
  end for
  set index the private cloud ID with max priority
  for each cloud u_i ∈ PUB do
    if u_i is not alarmed then
      calculate cloud priority η_i
    end if
  end for
  set index the public cloud ID with max priority
  for each job J_i(t) ∈ [J_1(t), J_2(t), ... do
    if key Equals 0 then
      Dispatch the job J_i(t) to cloud u_index
    else
      Dispatch the job J_i(t) to cloud r_index
    end if
  end for
end for
```

3.4.3 Cloud scaling

We set two thresholds for scaling up and down with a lower bound and an upper bound, denoted as σ_l and σ_u, respectively.

$$\sigma_i(t) = max\{\frac{\lambda_i^{CPU}(t) - \sum_{n=1}^{N(t)} d_{in}^{CPU}}{\lambda_i^{CPU}(t)}, \frac{\lambda_i^{NET}(t) - \sum_{n=1}^{N(t)} d_{in}^{NET}}{\lambda_i^{NET}(t)}, \frac{\lambda_i^{MEM}(t) - \sum_{n=1}^{N(t)} d_{in}^{MEM}}{\lambda_i^{MEM}(t)}, \frac{\lambda_i^{DISK}(t) - \sum_{n=1}^{N(t)} d_{in}^{DISK}}{\lambda_i^{DISK}(t)}\} \quad (3)$$

We use Eq. 2 to calculate the priorities of public clouds. Here, we use Eq. 3 to get the resource usage of the current cloud in simulation experiments. During each time slot, the load balancer checks whether each cloud needs to adjust its scale. In real world, cloud provider provisions API to get the states of cloud resources, which is much preciser. Algorithm 2 describes the resource allocation process.

Algorithm 2 Cost-Effective Resource Allocation

```
set upscale = true, downscale = false
for each time slot t ∈ [0, 1, 2, ...] do
  for λ_i ∈ PUB do
    if σ_i(t) ≤ σ_u then
      upscale = false
    end if
    if σ_i(t) ≤ σ_l then
      downscale = true
    end if
  end for
end for
if upscale is true then
  send scale − up direction to the prior cloud
end if
if downscale is true then
  send scale − down direction to the worst cloud
end if
```

3.5 Discussion

In this section, we design an LCC load balancing algorithm to implement job scheduling. This problem is an NP-hard optimization problem. To get the optimization solution may spend a lot of time. Considering in a large-scale distributed network environment, network delay plays a important role in system performance.

4 Experiment

In this section, we conducted a series of simulation experiments from different perspectives. The results demonstrate that our load balance mechanism for hybrid cloud application could reach cost-effective, meanwhile it also provides information on the pros and cons.

4.1 Data preparation

Requests: We first construct the realistic job requests according to the data traces obtained from Google data center, which include the job arrive/leave time and resource cost information. Then we simulate another three specific application requests which have high demand on CPU, memory, and disk respectively.

Clouds: As the requests vary in Google trace, requests should be sent to the suitable cloud for handling. So using multiple public clouds is better than single public cloud. As an instance's startup need some delay, using multi-cloud can somehow reduce the delay. We use three public clouds and two private clouds to construct a hybrid cloud. The parameters of each instance on each cloud is set according

Table 2 Parameters of public clouds

Cloud ID	CPU	MEM	DISK	Price
Cloud 1	8	8	20	0.628
Cloud 2	2	8	20	0.375
Cloud 3	2	3	80	0.453

to the typical instance types of Amazon EC2 and Google Computing Engine.

We list the parameters of public clouds in Table 2. In the price column, the pricing details are obtained in providers' websites. We also list the resource demand of the three types of jobs used in experiments in Table 3.

4.2 Experiment results

4.2.1 Function test

In this experiment, we show functions that the load balancer can reach. The system able to guarantee the private cloud resource use first, keep private cloud at a higher resource utilization. The system will choose most suitable cloud for out-sourcing according to the request type. Further more, it is responsible to the price changes keeps global load balanced.

Figure 2 demonstrates respective CPU usage of the private and public clouds in a hybrid cloud. As shown in Fig. 2, the average CPU usage of VMs in the private cloud stays above 0.8 in the whole process except five short time periods. Such a phenomenon indicates that our algorithm works based on our design concept. As mentioned in Section 3, to maintain the least cost, we use the private cloud as much as possible while outsource excessive workloads to the public cloud. Hence, we need to ensure the average CPU usage of VMs in the private cloud stays in a range stably. Meanwhile, the average usage of VMs in the public cloud fluctuates wildly, which is led by bursty workloads. Although the workloads is bursty, the CPU usage of private cloud is stable, which further demonstrates the effectiveness of our algorithm.

Figure 3 demonstrates the trends of the number of running VMs in three deployed clouds under different types of workloads. In the first time period, the workloads are CPU-intensive. The scale of Cloud 1 increases to the largest for it

Table 3 Requirement of resources

APP	CPU	MEM	DISK
CPU_APP	0.65	0.2	1.5
MEM_APP	0.2	0.68	2.5
DISK_APP	0.2	0.18	4.5

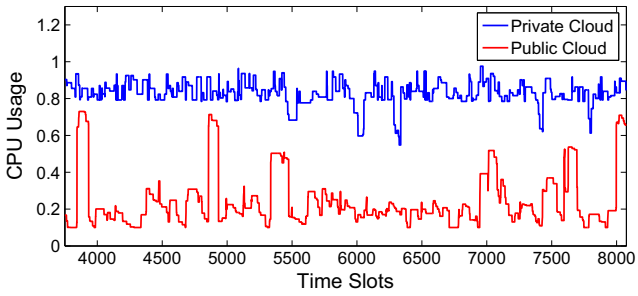

Fig. 2 CPU usage of the private and public clouds in a hybrid cloud

is good at processing CPU-intensive workloads. Then, when the type of workloads switch to the memory-intensive type, the scale of Cloud 1 goes down while the scale of Cloud 2 ramps up to the largest. Finally, the same trend can be observed in the third time period. As a result, our algorithm is sensitive about the changes in types of workloads and can adjust the scales of public clouds based on it.

Figure 4 shows the scales of public clouds when their prices change. In Fig. 4, the scale of Cloud 2 increases after its price goes down. Meanwhile, when the price of Cloud 1 increases, its scale decreases a little. The effects indicate that our algorithm can adapt the price volatility and choose the cost-effective clouds adaptively.

4.2.2 Performance analysis

For comparisons, we use random algorithm which is most commonly used on load balance problems.

Figure 5 plots the effects brought by different strategies of deploying the public cloud. In Fig. 5, the average delay processed by single cloud is shorter than that processed by multiple clouds. Furthermore, the cost of deploying single cloud is less than that of deploying multiple clouds. Hence, by deploying multiple public clouds, we can provision high quality services as well as saving cost. Meanwhile, to show the effectiveness of LCC, we compare LCC to a random strategy. As plotted in Fig. 5, the delay of the random strategy is longer than LCC's while the cost is more than LCC's.

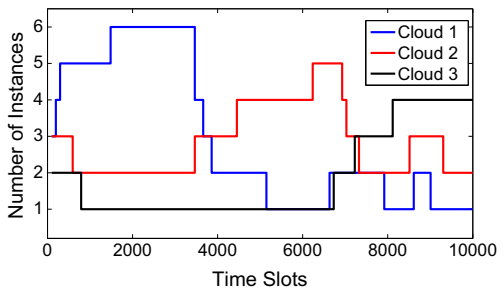

Fig. 3 Resource requirement trace of CPU-intensive, memory-intensive, and disk-intensive jobs

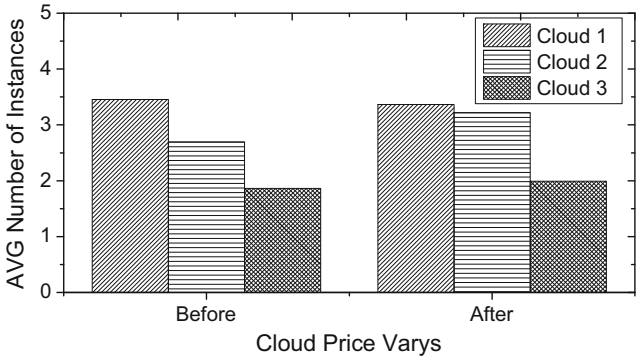

Fig. 4 The price of Cloud 1 increases 0.1 dollar while the price of Cloud 2 decreases 0.1 dollar during the optimization

As a result, our algorithm works well in the hybrid cloud environment.

Figure 6 shows different scales of public clouds in the hybrid cloud. As shown in Fig. 3, the average number of VMs in Cloud 1 is the largest when the workloads are CPU-intensive. Then, the number in Cloud 2 is the largest when the workloads are memory-intensive while the number in Cloud 3 is the largest when the workloads are disk-intensive. Such phenomenon indicates that our algorithm is aware of the types of workloads. Furthermore, our algorithm can adjust the scales of deployed clouds based on different types of workloads and make the best use of the public clouds.

Figure 7 plots the effects brought by different numbers of public clouds. As shown in Fig. 7, when the number of public clouds increases from 1 to 4, the values of latency and cost fall sharply. When deploying more clouds, our algorithm can adjust the respective scales as well as distribute workloads to provision cost-effective services. We have thought that the principle was "the more choices, the better", adding cloud provider only bring less cost or not, no higher. Because the load balancer can choose whether to use the cloud or not. However, when the number of public clouds

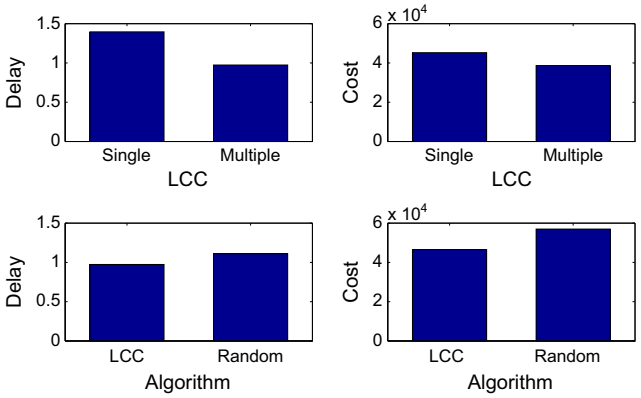

Fig. 5 Comparison test on different public clouds and comparison test between LCC and random algorithm

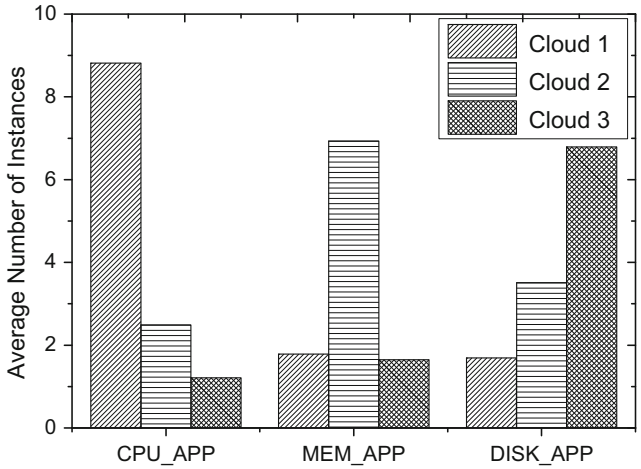

Fig. 6 The average number of instances which three applications use on three different public clouds

increases from 4 to 7, the latency decreases slightly while the cost increases markedly. Since we need to maintain the least scale of each public cloud, deploying more clouds does not contribute to cost-saving. Furthermore, the capacity of hybrid cloud is too large to improve performance.

5 Related work

Load balancing problem [8] can be divided into two sub-problems: job request scheduling and dispatching and migration of resource. The task scheduling algorithms can be broadly classified into two types: static algorithms and dynamic algorithms. The commonly used algorithms contain randomized, round robin, min-min scheduling algorithm, opportunistic load balancing and so on.

Traditional cloud load balancing service is one of tools which helps cloud providers to meet the QoS requirements of cloud users, and maximize provider's profit by optimizing the resource utilization. However, in recent years, especially when the hybrid cloud becomes popular, more

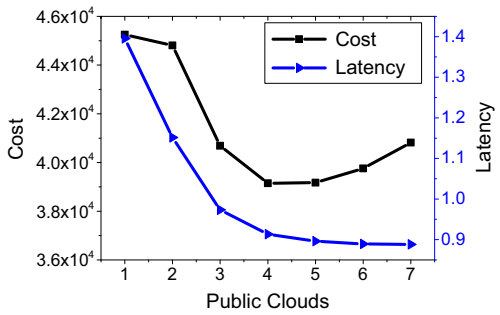

Fig. 7 Measurement on the global average cost and delay under the number of public clouds varying form 1 to 7

and more cloud users want to have more initiatives on controlling their cloud resources.

Li et al. [6] focus on classifying and measuring the typical services which public clouds commonly provide. They develop a benchmark tool called CloudCmp to help cloud users comparing different clouds' performance and price. Nevertheless, the cloud price may change, CloudCmp cannot respond to real-time fluctuation of the price. Since current public cloud providers provide various types of instances [1], cloud users are facing the dilemma of hard choices [6]. Zhao *et al.* propose an online algorithm for joint VM pricing, job scheduling and server provisioning in a cloud supported by geo-distributed datacenters in [12].

[11] gives out a hybrid cloud model which uses a workload factoring scheme to separate base workloads and flash crowd workloads. It proactively pushes the flash crowd workloads onto the public cloud and leaves rest in the private cloud, so as to achieve resource efficiency and reduce data replication. The authors design an online algorithm for optimal request distribution to meet cloud bursting for the hybrid cloud.

Based on former works [6, 12], we propose a two-tier load balancing mechanism which provisions VM-level cost-effective load balancing for hybrid cloud applications, which brings considerable benefit and performance for cloud users.

With respect to studies on managing the performance overhead of VMs, part of the work in [9] summarizes it under diverse scenarios of the IaaS cloud. Li et al. in [6] focus on classifying and measuring the typical services which IaaS public clouds provide. Complementary to [9] and [6], we deal with performance issues of multiple public clouds in hybrid cloud scenarios. In terms of addressing network performance, Yi et al. take a close look at the unique challenges in building a network highway for big data in [10]. Complementary to [10], we consider network performance of web services when scheduling requests.

[11] gives out a hybrid cloud model which uses a workload factoring scheme to separate base workload and flash crowd workload. Furthermore, [7] models e-commerce web services and proposes an online algorithm to address the load balancing problem under flash crowds in hybrid cloud scenarios. Inspired by [7] yet different from it, we address the load balancing problem with considering multiple public clouds.

6 Conclusion

This paper proposes a Least Cost per Connection (LCC) load balancing algorithm for hybrid cloud applications, which helps provide cost-effective services. Moreover, we design a system prototype for simulation experiments. The results show that our system can guarantee the private cloud usage as well as achieve the goal of high performance with low cost. Compared with the single cloud strategy, the cost and latency of our system decrease 30.2 and 10.1 percent, respectively. This system can be integrated into the hybrid cloud management platform like CloudForms, ManageIQ and so on. Other works we are doing now are making this system a plug-in feature into ManageIQ [3].

Acknowledgments The research was supported by a grant from the National Natural Science Foundation of China (NSFC) under grant No. 61520106005.

References

1. Auto Scaling. http://aws.amazon.com/autoscaling/
2. Google Computing Engine. https://cloud.google.com/compute/pricing
3. ManageIQ Open Source Project. http://manageiq.org/
4. OpenStack. http://www.openstack.org/
5. (2015) State of the cloud report. Report
6. Li A, Yang X, Kandula S, Zhang M (2010) Cloudcmp: Comparing public cloud providers. In: Proc. of IMC 2010
7. Niu Y, Luo B, Liu F, Liu J, Li B (2015) When hybrid cloud meets flash crowd: Towards Cost-Effective service provisioning. In: Proc. of IEEE INFOCOM
8. Shaw S, Singh A (2014) A survey on scheduling and load balancing techniques in cloud computing environment. In: Computer and Communication Technology (ICCCT), 2014 International Conference on, pp 87–95. doi:10.1109/ICCCT.2014.7001474
9. Xu F, Liu F, Jin H, Vasilakos A Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions. Proceedings of the IEEE 102(1), 11–31
10. Yi X, Liu F, Liu J, Jin H (2014) Building a network highway for big data: architecture and challenges. Netw, IEEE 28(4):5–13
11. Zhang H, Jiang G, Yoshihira K, Chen H (2014) Proactive workload management in hybrid cloud computing. IEEE Trans Netw Serv Manag 11(1):90–100
12. Zhao J, Li H, Wu C, Li Z, Zhang Z, Lau F (2014) Dynamic pricing and profit maximization for the cloud with geo-distributed data centers. In: INFOCOM, 2014 Proceedings IEEE, pp 118–126