Enabling Customer-Provided Resources for Cloud Computing: Potentials, Challenges, and Implementation

Haiyang Wang, *Member, IEEE*, Feng Wang, *Member, IEEE*, Jiangchuan Liu, *Senior Member, IEEE*, Dan Wang, *Senior Member, IEEE*, and Justin Groen

Abstract—Recent years have witnessed *cloud computing* as an efficient means for providing resources as a form of utility. Driven by the strong demands, industrial pioneers have offered commercial cloud platforms, mostly datacenter-based, which are known to be powerful and effective. Yet, as the cloud customers are pure consumers, their local resources, though abundant, have been largely ignored. In this paper, We present `SpotCloud`, a real working system that seamlessly integrates the customers' local resources into the cloud platform, enabling them to sell, buy, and utilize these resources. We also investigate the potentials and challenges towards enabling customer-provided resources for cloud computing. Given that these local resources are highly heterogeneous and dynamic, we closely examine two critical challenges in this new context: (1) How can the customers be motivated to contribute or utilize such resources? and (2) How can high service availability be ensured out of the dynamic resources? We demonstrate a distributed market for potential sellers to flexibly and adaptively determine their resource prices through a repeated seller competition game. We also present an optimal resource provisioning algorithm that ensures service availability with minimized lease and migration costs. The evaluation results indicate it as a flexible and less expensive complement to the pure datacenter-based cloud.

Index Terms—Cloud computing, customer-provided resource, measurement

1 Introduction

Recent advances in cloud computing offers an efficient means for providing computing as a form of utility. Such enterprise cloud providers as Amazon, Google, and Microsoft have enjoyed significant increase of their customer populations, enforcing them to constantly upgrade and expand their datacenter infrastructures.[1] Yet, the existing enterprise cloud capacity is still a fraction of the need when we consider the fast growth of the customers' demand. Recent studies suggest that the user experience of enterprise clouds, such as Amazon EC2, is indeed decreasing.[2]

On the other hand, the customers' local computing resources are still rapidly evolving. Today's advanced consumer CPUs, like the Intel's Core i7, is no slower than many of the server CPUs a few years ago; this CPU is even faster than most of the medium or even some large instances in today's enterprise cloud. The aggregated computation, storage, and network resources available at cloud customers are indeed more than that at a typical datacenter. In other words, the cloud as an elusive platform is not simply due to the abundant resources available at a remote location; yet meeting resource demand is a key factor to the cost of maintaining datacenters and providing cloud services, and the resulting service prices often hinder customers from migrating to the cloud.[3]

There have been recent studies on smart service partitioning that keeps certain tasks local [1]. We however envision a more general solution that seamlessly integrates the customers' local resources into the cloud platform, enabling them to sell, buy, and utilize these resources, thus offering a more flexible and less expensive complement to the pure datacenter-based cloud.

In this paper, we take a first step towards the feasibility and the system design of enabling customer-provided resources for cloud computing. We present the detailed design of `SpotCloud`,[4] a real working system that enables customers to seamlessly contribute their resources to the

1. http://blog.rightscale.com/2009/10/05/amazon-usage-estimates/.
2. http://www.thebuzzmedia.com/amazon-ec2-performance-drops-too-many-users/.
https://www.cloudkick.com/blog/2010/jan/12/visual-ec2-latency/.

• H. Wang is with the School of Computer Science, University of Minnesota at Duluth, 1114 Kirby drive, Duluth, MN 55812-2496. E-mail: haiyang@d.umn.edu.
• F. Wang is with the School of Computing Science, University of Mississippi, University, MS 38677. E-mail: fwang@cs.olemiss.edu.
• J. Liu is with the School of Computing Science, Simon Fraser University, Burnaby, Burnaby, BC V5A 1S6, Canada. E-mail: jcliu@cs.sfu.ca.
• D. Wang is with the Department of Computing Science, Hong Kong Polytechnic University, Hung Hom, Kowloon 0000, Hongkong. E-mail: csdwang@comp.polyu.edu.hk.
• J. Groen is with Enomaly Inc., Toronto, ON, Canada. E-mail: justin@enomaly.com.

Manuscript received 5 Oct. 2013; revised 31 May 2014; accepted 10 June 2014. Date of publication 15 July 2014; date of current version 5 June 2015. Recommended for acceptance by X. Liu.
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.2014.2339841

3. To put things into perspective, leasing the same amount of computation and storage resources from Amazon EC2 as that of an ordinary PC now costs $910 per year (with *Reserved large Instances*), which is not cheaper than purchasing the PC.
4. http://www.spotcloud.com/.

overall cloud. Since its deployment in November 2010, SpotCloud has attracted customers worldwide. In this paper, we overview the SpotCloud design and, through trace-analysis, demonstrate it as a promising complement to the datacenter-based cloud. In particular, it offers cloud service with flexible and relatively lower cost and yet comparable performance to state-of-the-art enterprise clouds.

Given that these local resources are highly heterogeneous and dynamic, we closely examine two critical challenges in this new context: (1) *How can the customers be motivated to contribute or utilize such resources?*and (2) *How can high service availability be ensured out of the dynamic resources?* We demonstrate a distributed market for potential sellers to flexibly and adaptively determine their resource prices through a repeated seller competition game. We also present an optimal resource provisioning algorithm that ensures service availability with minimized lease and migration costs. We further examine the lease and migration costs in the presence of dynamic resource availability in the real deployment, and highlight the trade-offs therein.

The rest of this paper is organized as follows: In Section 2, we present the related works. Section 3 discuss the framework design as well as the challenges. After that, we examine the pricing problem and the resource provisioning problem in Sections 4 and 5, respectively. Section 6 presents the SpotCloud design and its performance result. We further investigate the cost and availability issues in the system in Section 7. Finally, Section 8 concludes the paper.

2 Related Works

The salient features of cloud computing have enticed a number of companies to enter this market [2], [3], [4], [5], [6] and have also attracted significant attention from academia [1], [7], [8], [9]. There have been a series of measurement and comparison of the diverse cloud platforms. Garfinkel studied the performance of Amazon's Simple Storage Service (S3) and described the experience in migrating an application from dedicated hardware to S3 [10]. Walker investigated the performance of scientific applications on Amazon EC2 using both macro- and micro-benchmarks [11]. A recent study by Li and Yang [12] further presented CloudCmp, a systematic comparator for the performance and cost of cloud providers in today's market. These studies have mainly focused on cloud enabled by enterprise datacenters. They have demonstrated the power and benefit of such enterprise clouds, but also revealed many of their weaknesses. In particular, Ward [13] showed that the virtualization technique in Amazon EC2 can lead to dramatic instabilities in network throughput and latency, even if the datacenter network is only lightly loaded. To rebalance the workloads across physical machines, VM migration is also widely suggested [14]. A recent study by Shrivastava et al. [15] introduced an application-aware migration approach which can greatly reduce the network traffic during the VM migration.

There have been recent studies on the partition and allocation of services to the datacenters and the local computers, respectively [16], [17], [18], [19]. Our work differs from them through seamless provisioning the customer local resources to the overall cloud. There have been a significant amount of related works on capacity provisioning in computer clusters

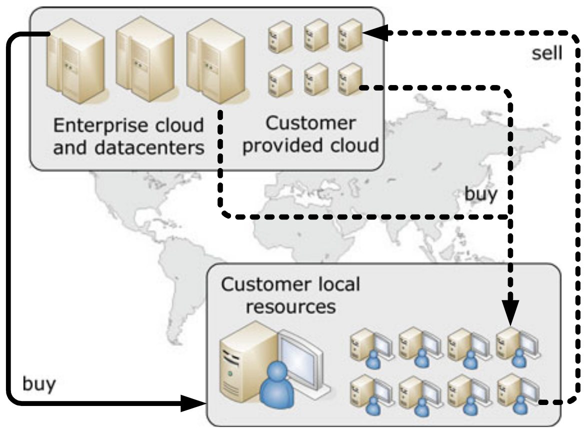


Fig. 1. Overview of framework.

and grids [20], [21], [22]. The classical feedback control theory has also been used to model the bottleneck tier in web applications [23], [24]. Based on these studies, Sharma et al. [17] further proposed a cost-aware provisioning algorithm for cloud users. Utilizing customer-provided resources, however, presents new challenges given their stronger dynamics.

Our work is related to peer-to-peer, which also seeks to utilize local user resources. Yet most of the peer-to-peer systems have focused exclusively on content sharing [25], while a cloud platform is expected to offer diverse resources for a broad spectrum of services. Moreover, though certain incentive mechanisms have been developed for peer-to-peer systems to penalize free-riders, there still lacks clear business/pricing models to enable enterprise-level services, not to mention those demanding high availability as we are targeting. Note that our system is a commercial cloud platform. It is different from such educational platforms as Planet-lab [26] which hardly be accessed by the general public.

3 Enabling Customer Resources for Cloud

3.1 Framework Design

We now offer an overview of our framework that enables customer-provided resources in the cloud. We will then address the key design issues in this framework, and present a practical system implementation, namely Enomaly's `SpotCloud`, along with its performance measurement.

In Fig. 1, we outline the relation between the cloud providers and their customers. The solid lines illustrate the business model for the existing cloud, with the customers being pure resource-buyers. As such, their local resources have been largely ignored or exclusively used for each individual's local tasks, which are known to be ineffective. Aiming at mitigating this gap between centralized datacenter resources and the distributed local resources, our framework enables individual cloud customers to contribute their otherwise exclusively owned (and idled) resources to the cloud and utilize others' resource if needed, as illustrated by the dotted lines in the figure.

3.2 SpotCloud: A Practical System Implementation

In this part, we will present a real-world system implementation, namely Enomaly's `SpotCloud`, which further addresses a series of practical challenges. As shown in

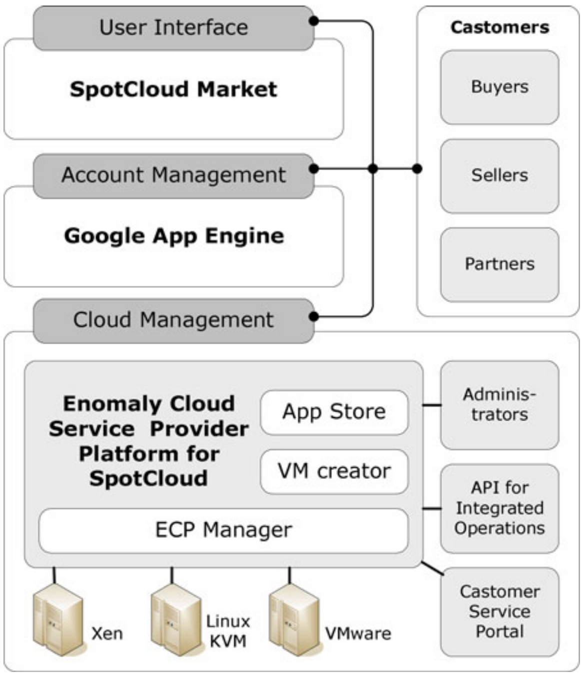


Fig. 2. Software module design.

Fig. 2, SpotCloud consists of three key modules: cloud management, account management, and user interface. The cloud management module supports a variety of common *hypervisors* (also known as *virtual machine managers*) including Xen, KVM and VMware as well as a highly fault tolerant and distributed *Extensible Messaging and Presence Protocol* (XMPP) with built-in failover capabilities. It also works with our resource provisioning algorithm for VM provision and migration. The account management, built on the Google App engine, allows the customers to create Google accounts for the SpotCloud marketplace. This marketplace is provided by the user interface module to let the potential sellers post and update their configurations and prices for the contributed resources. Typical applications running in SpotCloud are as follows:

- Testing and Development Platforms
- Regional Load Balancing
- Image and Video Processing
- Scientific Research Data Processing

Request sent by SpotCloud:

```
https://api.provider.com/utilization?
login=Login
&ecp_username=39480304
&ecp_auth_digest=
lfOBcOAfcLPqPUz1b1dE4MYQFSw=
```

Response returned by resource sellers:

```
{
   total_memory: 4085,
   free_storage: 84146,
   free_memory: 1397,
   total_storage: 291618,
   loadfifteen: 1.7
}
```

Fig. 3. An example of message format for utilization monitoring.

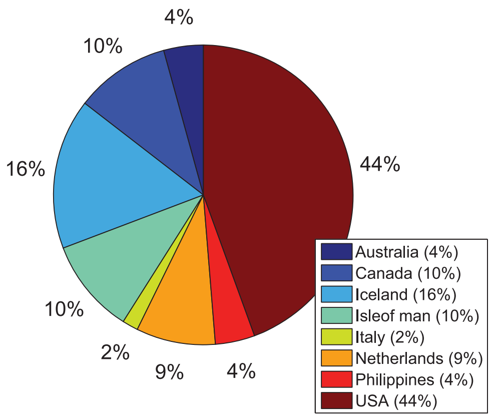


Fig. 4. Locations of SpotCloud resources.

- Financial Modeling and Analysis
- Content Distribution

Fig. 5 shows a simplified finite-state machine (FSM) in the SpotCloud system, where the *Authentication* state is managed by the account management module; the *Wait*, *Open* and *Billing* states are managed by the user interface module, and the rest of states are managed by cloud management module. The dark circles refer to the states that are used to communicate with the sellers. In particular, SpotCloud uses a set of *RESTful* (wait for sellers' information/reply to go to the next state) HTTP-based APIs for such communications. Fig. 3 shows an example of the message format when SpotCloud sends a *HTTP utilization monitor request* to a seller, where *loadfifteen* field includes the average load over the past 15 minutes for the seller. More details can be found in our API and Third Party Provider Integration Guide [27].

As shown in Fig. 4, SpotCloud platform has already attracted the Internet customers worldwide. We now examine a sample set of 116 typical customers for a basic understanding of the system performance and efficiency.

We first check the number of CPUs in the SpotCloud VMs. As shown in Fig. 6, it is easy to see that most SpotCloud resources ($>75\%$) possess less than four virtual cores. Each virtual core provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron processor. This is not surprising since most of the customer-provided resources are

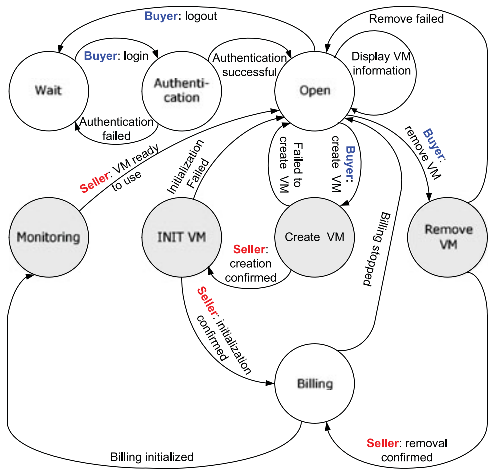


Fig. 5. Finite-state machine in SpotCloud.

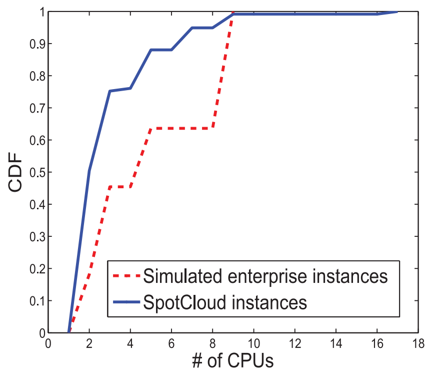


Fig. 6. # of CPUs.

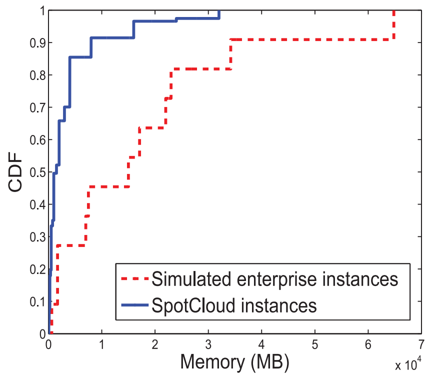


Fig. 7. Memory size.

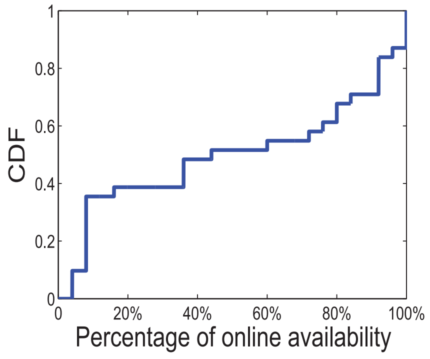


Fig. 8. Online availability.

not as powerful as those from enterprise datacenters.[5] Yet, there are also some relatively powerful VMs; for example, a customer-provided VM has 16 virtual cores with two computation units in each core, which is capable of running certain CPU intensive tasks. We also show the memory sizes on the VMs in Fig. 7. We can see that most VMs (80 percent) in SpotCloud have a memory less than 5 GB, which is not extra huge but is suitable to run most of the real-world tasks. It is worth noting that, the curves of SpotCloud VMs are quite smooth, indicating the existence of more flexible options to meet the heterogeneous demands from customers.

5. This data is obtained by crawling the websites of enterprise cloud service providers. For example, the VM capacity information from Amazon and Rackspace.

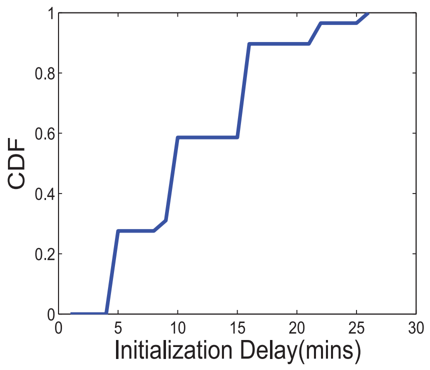


Fig. 9. Initialization delay.

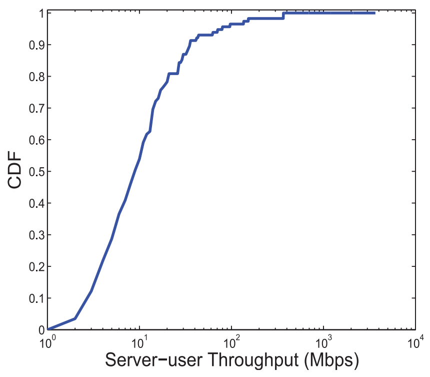


Fig. 10. Server-client throughput.

Different from enterprise servers that are known to have very high availability, the resource availability in SpotCloud is mostly depending on the sellers. Fig. 8 shows the online availability of the resources for one month. It is easy to see that the instance availability in SpotCloud is persistent: 40 percent VMs have an online availability below 20 percent, that is, less than six days in the 30-day measurement period. This availability is acceptable for short-term tasks lasting for a few hours or days. For longer tasks, SpotCloud has to carefully assist its buyer to choose proper VMs based on our proposed resources provisioning algorithms.

It is worth noting that before a buyer can really use a cloud instance, there is a delay due to the necessary initialization processes in any cloud platform. For example, the AWS Management Console [28] shows that it generally needs 15 to 30 minutes to initialize a Windows instance on Amazon EC2 before a customer can really connect to it. For SpotCloud, as shown in Fig. 9, we can see that most VMs (more than 60 percent) can be initialized within 10 minutes, and the maximum initialization delay is less than 27 minutes. This is considerably lower than that of Amazon EC2. The reason is that the system/user profiles of SpotCloud VMs are already included in buyers' VM appliances. Note that VMs' operation systems can also be personalized by the buyers in SpotCloud. Yet, if buyers do not want to decide the OS type, Linux (Ubuntu 10.10) is set as a default, which indeed has even lower initialization delay. We further investigate the VM throughput. As shown in Fig. 10, we can see that the throughput of over 50 percent VMs are more than 10 Mbps, which is good enough to deliver customers' contents to the cloud servers in normal cases.

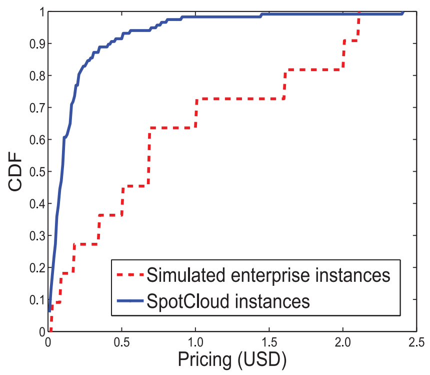


Fig. 11. Pricing distribution.

One important feature of cloud services is that the customers pay only for what they have used. In most of the cloud systems, this cost consists of two major parts: The cost of leasing VMs, and the cost of data transfer. Their pricing model is computed/decided carefully by the enterprise service providers. As we have discussed in the previous section, the price of SpotCloud VMs, however, is customized by individual sellers who provide/sell their cloud capacities. As shown in Fig. 11, we can see that the SpotCloud VMs are mostly very cheap. Moreover, this curve is also quite smooth indicating that the buyers have very high flexibility in selecting VMs in this customer-provided cloud platform.[6]

4 Pricing with Heterogeneous Resource Providers

Our trace analysis indicates the efficiency of our system model design. In particular, SpotCloud has attracted many customers to contribute their local resources in our marketplace.

It is easy to see that one critical challenge is to offer enough incentive for a customer to contribute her/his resources or utilize others'. The problem is further complicated given that the customers are highly heterogeneous, making the coarse-grained pricing model used by the existing cloud providers hardly working. In our system design, we address this problem through a distributed resource market that allows the customers to decide the quality, quantity, and pricing of their local resources to be contributed. We therefore demonstrate the effectiveness of our market model with a repeated competition game design.

4.1 Modeling of Resource Pricing

In most of the existing enterprise cloud platforms, fixed pricing remains the most popular strategy. Amazon EC2, as a typical example, advertises $0.02-2.62 per hour for each of its *On Demand Virtual Machine* instances, depending on their types. Recently, dynamic pricing also been introduced, e.g., the "spot pricing" in EC2 [30] that aims at better utilizing the vacant capacities in the datacenters. It is known that the spot price will be dynamically adjusted to matching the supply and demand, though the full details have not been disclosed.

6. In terms of the system applicability, the study in [29] also presents a case study when we use Spotcloud to assist such content delivery applications as BitTorrent. In particular, the SpotCloud VMs can serve as temporary peers to enhance the downloading performance for the P2P users. It shows that the customer-provided resource is efficient to support short-term applications.

TABLE 1
List of Notations

N	Total number of sellers
I	The set of VM sellers where each seller $i \in I$
x	Vector of buyers' demands on sellers
c	Vector of sellers' capacities
p	Vector of sellers' prices
q	The quantity actually sold capacities
γ	Vector of sellers' cost factors
$\overline{S}$	Strategy set a cross N sellers i
S_i	The set of strategies for seller i
t	Time slots
$Z(t)$	Strategy vector at time t
D	Amount of user demand faced by seller
δ	Discount factor

Since the potential resource providers in SpotCloud are heterogeneous and are not forced to contribute their resources, a working business model is necessary to offer them enough incentive. Therefore, instead of setting a standardized pricing rule for unit resource, we suggest a distributed market that allows the potential providers (i.e., *sellers*) to decide the quality, quantity, and pricing of their local resources to be contributed, in which:

1) A seller will advertise the configuration (amount and availability) of its local resources as well as the asking price; such information will be seen by other sellers and buyers;
2) Both resource sellers and buyers are rational: given the advertised prices, a buyer will try to minimize the cost for resource provisioning, and a seller will try to maximize the profit;
3) After seeing others' advertised information, a seller will adjust her/his own configuration and price to maximize her/his potential profit.

The intuition behind this design is that the sellers have better knowledge of their own resources in terms of both running costs and expected values. If they cannot find a good way to gain profits, any fixed or dynamic pricing rule will fail to give them the incentive to join cloud markets. This business model can be formulated as a variation of a *Repeated Seller Competition* game [31] as follows. To facilitate our discussion, we list the key notations that will be used in this paper in Table 1.

We start with a general model with N resource sellers. Each seller can be thought of as a cloud service provider operating s/he own local resources. We denote the vector of sellers' resource by $c = (c_1, \dots, c_N)$. Investing in resource is costly. In particular, the cost of sharing resource c_i for seller i is $\gamma_i c_i$ where $\gamma_i > 0$ for $i \in \{1, \dots, N\}$. We denote the price charged by seller i (per unit demand) by e_i and denote the vector of prices by $e = (e_1, \dots, e_N)$. On the other hand, the buyer demand is given by a set $W = \{w_1, w_2, \dots\}$, which may result from a continuum of consumers who demand exactly one unit up to their *reservation prices* (the unit price that expected by the buyers). Therefore, $|W|$ shows the aggregate demand in the system.

For the sellers in the system, their possible strategies are: (1) Provides c_i cloud resources with unit price e_i; (2) Inactive and not willing to contribute any local resources (n.a). This strategy space can be written as:

$$S_i := \{e_i, c_i\} \bigcup \{n.a\}, \tag{1}$$

where $\{e_i, c_i\}$ means that seller i is active and provides c_i cloud resources with unit price e_i, and $\{n.a\}$ means that seller i is not willing to contribute any local resources. The game is thus played as follows: each seller i announces a capacity c_i and a unit price e_i which it is willing to sell. In this case, its profit is given by:

$$\pi(S_i) = \begin{cases} e_i \cdot c_i - \gamma_i \cdot c_i - F & for \quad e_i \in [0, v], \\ 0 & else \quad when \quad S_i = n.a, \end{cases} \tag{2}$$

where F is the fixed cost (e.g. the deployment overhead) when sellers configure their local resources for cloud service.[7] v is the *choke off price* indicating the maximin price of the sellers; no demand will face to seller i if $e_i > v$. The objective of each seller is to maximize the profit π.

Note that, all buyers will place their orders with the cheapest sellers. These orders are fulfilled until the cheapest sellers' capacities are all exhausted. If not all orders can be fulfilled by the cheapest sellers, we assume that the buyers will be rationed based on the the Beckmann-rationing rule [32]. In particular, a random sample of consumers will be served by the cheapest sellers. The remaining (unserved) buyers will place their orders with the next cheapest sellers. This procedure is repeated until either all the customers are served or all sellers are exhausted. The demand faced by a seller i is given by:

$$H_i = \frac{max\{|W| \cdot \left(1 - \sum_{j:e_j<e_i} \frac{c_j}{|W|_{[w=e_j]}}\right), 0\}}{\sum_j I_{[e_i=e_j]}}, \tag{3}$$

where $|W|_{[w=e_j]}$ is the number of demands with reservation prices equal to e_j and $I_{[e_j=e_i]}$ is the number sellers with the prices equal to e_i. Thus we have $q_i = min\{H_i, c_i\}$. Since it is a repeated game, when we use $Z(t)$ to denote the strategy vector of period t , the expected payoff of seller i over all infinitely many periods is given by:

$$\sum_{t=1}^{\infty} \delta \pi_i[Z(t)], \tag{4}$$

where $0 < \delta < 1$ is a *discount factor*, denoting the sellers' patience (close to 0: not patient; close to 1: highly patient). In practice it denotes the probability of whether the sellers will have the chance to play the game in the future [33]. According to Folk Theorems [34], the optimal method of playing this repeated game is not to repeatedly play a Nash strategy of the stage games, but to cooperate and play a socially optimum strategy. We will thus discuss whether such a stationary outcome equilibrium exist in this repeated game.

Assume that $\overline{S}$ is a stationary outcome equilibrium set across N sellers in the repeated game with unit price e (where the sellers charge the same price per unit resource in each period). Since all sellers are active and the game is symmetric, we omit the index i where no confusion results. We further write the sellers' profit in this stationary equilibrium as follows:

$$\pi(\overline{S}) = \begin{cases} (e - \gamma_i) \cdot \frac{|W|_{[w \le e]}}{N} - F & if \quad \frac{|W|_{[w \le e])}}{N} < c, \\ (e - \gamma_i) \cdot c - F & else. \\ . \end{cases} \tag{5}$$

Since δ is a probability, we have $1 + \delta^2 + \delta^3 + \cdots = \frac{1}{1-\delta}$. Thus the expected profit (over infinitely many periods) at this stationary equilibrium is $\frac{1}{1-\delta}\pi(\overline{S})$. It is easy to see that the stationary outcome equilibrium will exist when $\frac{1}{1-\delta}\pi(\overline{S}) > 0$ Otherwise if $\frac{1}{1-\delta}\pi(\overline{S}) \le 0$, the sellers will prefer to be inactive ($n.a$) with zero profit. In other words, if there is a strategy set a cross N sellers such that $\pi(\overline{S})$ is positive, the stationary outcome equilibrium will always exist for δ sufficiently close to 1. Since the lower part if Eq. (6) is the upper bound of $\pi(\overline{S})$, it is not difficult to see that $\pi(\overline{S})$ is decreasing in N; it will further becomes negative for N sufficiently large due to the existence of F. Hence, we can give the upper bound of N in the system (the existence condition for the stationary equilibrium) as follows:

$$N^* = max\{N \,|\, \pi(\overline{S}) > 0\}. \tag{6}$$

Therefore, for all $2 \le N \le N^*$ there is always a stationary outcome equilibrium $\overline{S}$ for the system where the sellers charge the same price per unit resource in each period. An intuitive explanation for N^* is the maximum number of sellers in the cloud market. Moreover, if $\overline{S}$ is a stationary outcome equilibrium, it is easy to see that for any seller i $S_i \ne n.a$ (since the profit is larger than zero, the sellers will have no incentive to be inactive). Note that the unit price e for this stationary outcome equilibrium is not unique, and it is depending on the buyers' demand at price e: $L(e) = |W|_{[w \le e]}$. The bound of this price will be further discussed in our future works.

4.2 Discussions

We can learn that if we deploy a cloud market where the sellers can dynamically adjust their selling quantity and prices, the sellers will be able find suitable strategies to sell their resources for a higher benefit (instead of leaving the market which results to zero profit). In this business model, the sellers' incentive and their total population are both depending on the buyer demand. Large demands can give sharing incentive to more sellers (larger N^*) and smaller demands will reduce the number of N^*. Yet, unless $|W| = 0$, this business model can always give incentives to a given number of sellers to join the cloud market. Moreover, giving the infinite round of this game, the price per unit resource has the trend to converge to a stationary equilibrium [35]. Different with the fixed pricing models, this stationary price

7. Notice that if there are no positive fixed costs, selling nothing would yield zero profits. So the difference between activity and non-activity vanishes in that case, and the extension of the strategy space by the element "n.a." would not be necessary. More importantly, the configuration of local resources involves overheads in real-world; it is thus necessary to consider it in our model.

is dynamically decided and adjusted by the sellers based on buyers' demand. It is also worth noting that the pricing monopoly and cooperative cheating can hardly happen in such a system. The main reason is that there are also some other alternative cloud providers over the Internet such as EC2 and Google. If the sellers' prices exceed the unit prices in other platforms, the sellers' profits will also be decreased to zero. This feature was also captured by the choke off price v in Eq. (4).

It is worth emphasizing that we view customer-provided resources a complement to the datacenter resources, not a replacement. Given that the design and optimization of datacenter-based cloud have been extensively studied, in this paper, we will focus on the effective utilization of customers' local resources and their seamless integration into the overall cloud platform. While exploring distributed local resources have been closely examined in such other contexts as grid computing [36] and peer-to-peer [37], the state-of-the-art cloud environment poses a series of new challenges for our proposed solution. In particular, to enable enterprise-level services, we have to ensure high service availability when integrating customers' resources. Different from datacenters, there is no guarantee that a particular customer's local resources will be always online for cloud computing.[8] Fortunately, through trace-analysis and an adaptive algorithm design, we demonstrate that the stable service availability is possible with the distributed and dynamic resources.

5 Provisioning across Dynamic Customer-Provided Resources

We start from examining the problem of resource provisioning across dynamic customer-provided resources.

5.1 The Resource Provisioning Problem

We consider a generic model of N resource providers (instead of one giant provider as in the conventional cloud, i.e., the datacenter). Without loss of generality, we assume each provider only offers one virtual machine (VM) for the market, denoted by $S = \{s_1, s_2, \ldots, s_N\}$. The resource capacity of VM s_i includes computation power p_{s_i}, memory size m_{s_i}, disk size d_{s_i}, and network bandwidth b_{s_i}. Given that such a VM is available on the cloud platform only when the provider does not plan to use it locally, we use $A_{s_i}(t)$ to denote the availability of VM s_i at time t; that is, $A_{s_i}(t) = 1$ if VM s_i is available, and otherwise $A_{s_i}(t) = 0$. In practice, such information can be obtained by asking the provider to indicate when it offers the VM to the cloud platform.

For a customer that expects to lease resources from the cloud platform, her/his demands include the aggregate computation power, the aggregate memory size, the aggregate disk size, and the aggregate bandwidth, denoted by P, M, D and B, respectively. Such demands are also accompanied by a request period $[t_{start}, t_{end}]$ indicated by the customer.

Define a provisioning schedule as $W = \{(x_1, t_1, l_1), (x_2, t_2, l_2), \ldots, (x_k, t_k, l_k)\}$ $(t_{start} \le t_1 \le t_2 \le \cdots \le t_k \le t_{end})$, where each tuple (x_i, t_i, l_i) $(x_i \in S$ and $l_i > 0$ for $i = 1, 2, \ldots, k)$ means that, starting at time t_i, VM x_i is provisioned for period l_i. The problem is thus to find a proper provisioning schedule given the demands, subjecting to the following constraints:

1) VM Availability Constraint:

$$\forall (x_i, t_i, l_i) \in W,\ \forall t \in [t_i, t_i + l_i], A_{x_i}(t) = 1.$$

2) VM Utilization Constraint:

$$\forall (x_i, t_i, l_i) \in W,\ \ if\ \ \exists (x_j, t_j, l_j) \in W \text{ and } x_i = x_j,$$

$$\text{then } [t_i, t_i + l_i] \cap [t_j, t_j + l_j] = \emptyset.$$

3) Resource Requirement Constraint:

$$\forall t \in [t_{start}, t_{end}],$$

$$\sum_{i=1}^{k} p_{x_i} \cdot I_{[t \in [t_i, t_i + l_i]]} \ge P,$$

$$\sum_{i=1}^{k} m_{x_i} \cdot I_{[t \in [t_i, t_i + l_i]]} \ge M,$$

$$\sum_{i=1}^{k} d_{x_i} \cdot I_{[t \in [t_i, t_i + l_i]]} \ge D,$$

$$\sum_{i=1}^{k} b_{x_i} \cdot I_{[t \in [t_i, t_i + l_i]]} \ge B;$$

where $I_{[\cdot]}$ is the indicator function. The above constraints ensure the resource availability during the lease period, a VM can be leased only in one schedule at any time, and the resource demands are fulfilled at any time instance within the lease period, respectively.

Let c_{s_i} be the lease cost of VM s_i per unit time. Our objective is thus to minimize a cost function $f(W)$, which involves two parts in our scenario:

Lease cost. $\sum_{i=1}^{k} c_{x_i} \cdot l_i$, which is the total cost for leasing the VM in W.

Migration cost. $\sum_{t=t_{start}}^{t_{end}-1} \sum_{i=1}^{k} I_{[t_i + l_i = t]}$, which is the cost to migrate the service/data to a new VM should the old VM become unavailable. Given the dynamic resource availability, migration across different providers in the lease period is necessary in our system to ensure the demands are fulfilled within the whole lease period. Accordingly, it is calculated as the number of VMs that becomes unavailable before time t_{end}.

It is worth noting that, for a fully cooperative non-profit system, the costs here can simply be the provider's net costs for offering the services. Yet if the providers are profit-motivated, which is natural for a commercial system, the costs depend on the provider's expected resource prices, which we will examine in the next section.

8. The enterprise cloud service, such as Amazon EC2, is enabled by highly available dedicated datacneters. This is why the service availability was seldom considered in the existing cloud models.

```
Algorithm OptimalProvisioningSchedule()
1:  Sort $S$ with ascendant order based on rules 1-2;
2:  for $t \in [t_{start}, t_{end}]$, $Req[t] \leftarrow (P, M, D, B)$; end for
3:  Set $Stack$ empty; $k \leftarrow 0$; $time \leftarrow t_{start}$;
4:  $Cost \leftarrow 0$; $Cost_{min} \leftarrow \infty$; Set $Stack^*$ empty;
5:  while true,
6:    if $time > t_{end}$ or $Cost \geq Cost_{min}$,
7:      if $Cost < Cost_{min}$,
8:        $Cost_{min} \leftarrow Cost$; $Stack^* \leftarrow Stack$;
9:      end if
10:     goto 26;
11:   end if
12:   $k \leftarrow k + 1$;
13:   if $k > |S|$,
14:     goto 26;
15:   else if $A_{s_k}(time) = 0$,
16:     continue;
17:   end if
18:   Push $\{k, time, S\}$ in $Stack$;
19:   $Req[time] \leftarrow Req[time] - (p_{s_k}, m_{s_k}, d_{s_k}, b_{s_k})$;
20:   Update $Cost$ according to $f(W)$;
21:   if $Req[time] \leq (0, 0, 0, 0)$,
22:     $time \leftarrow time + 1$; $k \leftarrow 0$;
23:     Sort $S$ with ascendant order based on rules 1-3;
24:   end if
25:   continue;
26:   if $Stack$ is empty, break;
27:   else
28:     Pop $Stack$; Update $Req$ and $Cost$ accordingly;
29:   end if
30: end while;
31: Generate optimal schedule $W$ by $Stack^*$;
32: return $W$;
```

Fig. 12. Algorithm to compute the optimal provisioning schedule.

5.2 Optimal Provisioning with Dynamic Resources

The lease cost is a major concern in any cloud platform. Yet given the dynamically available resources across providers, the migration cost[9] is not negligible in our system, either. The availability requirement also introduces another dimension. As such, the solution space for the provisioning problem becomes much larger. We now present a branch-and-cut algorithm for optimizing a general cost function $f(W)$. We then present a series of heuristics to consider such overhead as migration costs and further minimizing the search space for online dynamic provisioning.

Our algorithm is summarized in Fig. 12. We first sort the VMs in S in ascending order of the lease cost per unit resource[10] (referred to as *rule 1*). For the VMs with the same lease cost per unit resource, we further order them in descending order on their first unavailable times after t_{start} (referred to as *rule 2*). This allows the VMs that are mostly available and with cheaper resources being explored first and near-optimal solutions can then be quickly found. With such solutions, we can further cut other search branches with equal or higher costs (lines 6-11) and greatly reduce the search space for the optimal solution. We also check whether current VM s_k is available at $time$ (lines 13-17). If not, we will skip the current one and go on to the next. Every time a VM s_k is selected (lines 18-20), it will be considered leased at $time$ and $Req[time]$ will be reduced by the VM's resource capacity $(p_{s_k}, m_{s_k}, d_{s_k}, b_{s_k})$, accordingly. After that, the algorithm checks if any other VM needs to be leased (lines 21-24). If not, $time$ will be increased by one unit and k will be reset to 0. In addition, similar to line 1, we sort the VMs in S based on rules 1-2, but then move the VMs used at previous time unit ahead to the beginning (referred to as *rule 3*). When the search finishes, the optimal provisioning schedule W will be generated and returned (line 31-32). This is quite similar to the classic *Interval Scheduling Problem* [38]. The difference is that our user demand is fractional and can be smoothly migrated between different VMs. Due to the sorting operation, this algorithm takes time $O(n\ log\ n)$ on inputs with n virtual machines.

9. In this paper the migration cost refers to the frequency of migration during the cloud deployment

10. For multiple types of resources, the most stringent one can be used for sorting.

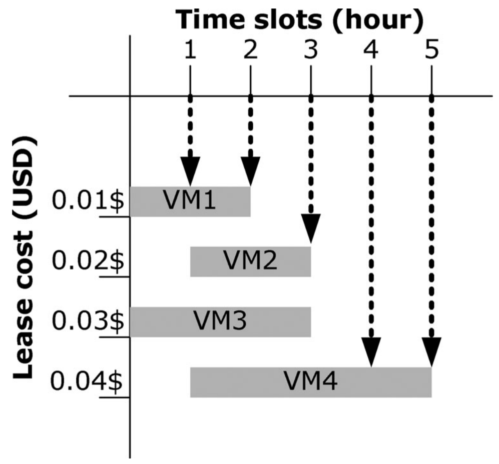


Fig. 13. An example of resource provisioning.

An example of lease cost minimization is shown in Fig. 13. In this case, the users need to make decisions across four VMs to support a five-hour task. We can see that for each time slot we search the available VM with minimum lease cost. In Fig. 13, we will use VM1 for the two hours, VM2 for one hour and VM3 for two hours. The optimality of this algorithm can be proved as follows:

Proof. We prove the optimality of this algorithm by contradiction. For a given time slot t, we assume that the task (at this time slot) is assigned to VMx in schedule W. Suppose VMy is a better assignment of this task with smaller lease cost at this time slot. Based on *rule 1* (all VMs in S are sorted in ascending order of the lease cost per unit resource), we can get that our algorithm already checked VMy before assigning the task to VMx. This is because the lease cost of VMy is smaller than the lease cost of VMx. However, the task is not assigned to VMy at time t. Based on Step5 to Step9 of Fig. 13, this means VMy is not available a this time slot (t). This leads to a contradiction because VMy is a valid assignment at time t. □

Depending on the application, lease cost and migration cost might have remarkably different contributions. We can then further speed up the online algorithm by assuming one of them is dominating. In particular, given a cost of interest, we can first sort the VMs in S by the rule (rule 1 or rule 2) corresponding to this cost. For the VMs tied with the rule for the cost of interest, we further sort them by the rule for the other cost. Then we pick the first k available VMs

TABLE 2
Smapled SpotCloud Instances

Price	vCPU	Memory	Storage	Country
\$0.002	1	256 MB	10 GB	Isle of Man
\$0.005	1	256 MB	15 GB	United States
\$0.010	1	512 MB	10 GB	United States
\$0.010	2	8 GB	10 GB	Isle of Man
\$0.019	1	256 MB	10 GBB	Netherland
\$0.020	1	4 GB	20 GB	United States
\$0.041	1	256 MB	15 G	Iceland
\$0.06	1	512 MB	30 GB	Poland
\$0.238	4	8 GB	50 GB	Iceland

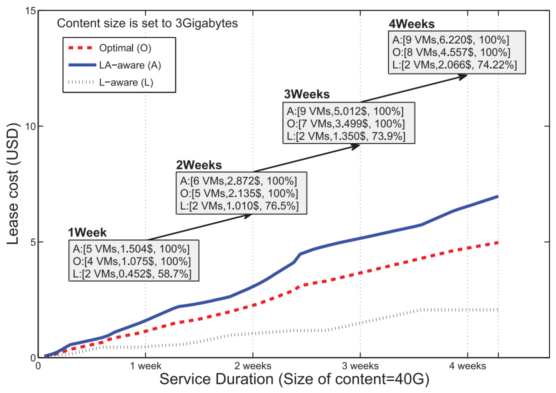
that can fulfill the demands at t_{start}. When a picked VM becomes unavailable, we pick a number of available but not yet used VMs from the beginning of S to fill up the demand gap until t_{end}. The effectiveness of the speedup algorithm as well as the tradeoffs between lease and migration costs will be evaluated in Section 7.

6 Performance Evaluation: Lease- versus Migration-Cost

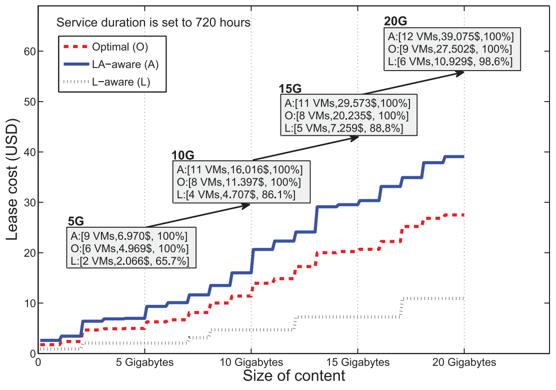
Given that the lease cost and migration cost play key roles in the resource provisioning and in pricing, we now take a closer look at their impact and tradeoffs with the SpotCloud trace data. We will also examine the effectiveness of heuristics respectively focusing on lease and migration costs, as discussed in Section 4. We call the algorithm that treats lease cost with more interest as *LA-aware*, and that treats migration cost with more interest as *MA-aware*. Note that both of them strike to ensure service availability out of dynamic customer-provided resources. For comparison, we also implement a state-of-the-art cost-aware-only algorithm (*L-aware*) [17], which however is difficult in ensuring service availability in SpotCloud as we will show. In the following discussions, *LA-aware* and *MA-aware* refer to our designs, and *L-aware* refers to the existing resource provisioning approach [17].

We apply the real data traces (as discussed in Section 3) from SpotCloud to run the algorithms. Table 2 shows a sampled set of the Spotcloud VMs. In this evaluation, the buyers will lease different VMs at different time to obtain service availability. This is different from the VM migration [15] because the relationship between VMs and physical machines are not changed. The selected Spotcloud VMs therefore consist of both KVM and XEN VMs. To better understand the performance of our heuristics, the optimal results are also enumerated as baselines for comparison.

From the traces, we find that the online patterns of customer-provided resources can be well fitted by a self-similar process with Hurst parameters [39] around 0.7 (more details can be found in our technical report [40]). We will be focusing on an online storage application, which, as compared to CPU-intensive applications, creates more challenges when resources are distributed at diverse locations. Yet, with certain modifications (mostly simplifications), our experiments and conclusions can be extended to CPU-intensive applications. This storage service enables a buyer to lease a set of SpotCloud resources to store her/his data contents; both the content size and the service



(a)



(b)

Fig. 14. Lease cost analysis with different: (a) service durations and (b) content sizes.

duration will be dynamically adjusted in our experiment. Note that we do not provide the cost comparison with enterprise cloud services such as Amazon S3. This is first because SpotCloud system is designed to complement the enterprise cloud services but not to replace them. Second, the pricing model of Amazon S3 is also largely different from SpotCloud; for example, besides the lease cost, S3 will also charge the customers based on the number of request and the amount of their data transfer. As such, a direct comparison can be quite difficult. Yet it is possible for a user to take advantage of both systems by splitting the storage, which can be an interesting direction worthy of further investigation.

L-aware versus LA-aware. We consider C-aware and LA-aware as the methods of resource provisioning. Fig. 14a shows the results when buyers use SpotCloud resources to have a storage for a 3 Gigabytes content with different service durations. The boxes in the figure show the number of used VMs, the lease cost, as well as the percentage of service availability. It is easy to see that if we apply the existing L-aware algorithms for customer-provided resources, the buyers will suffer from low service availability, which will be around 50 to 70 percent only, depending on the service duration. This service availability is obviously unable to support most Internet storage services. Fortunately, the proposed LA-aware algorithm provides very stable service availability, even when the buyers want to deploy this service for a long time, around 720 hours. As a trade-off, the buyers have to pay more to enable their service on more VMs. The lease cost of the LA-aware algorithm is also quite

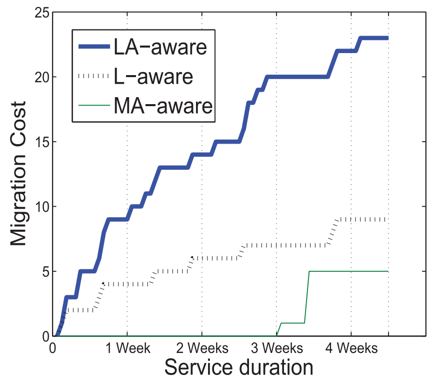


Fig. 15. Migration cost with different service durations.

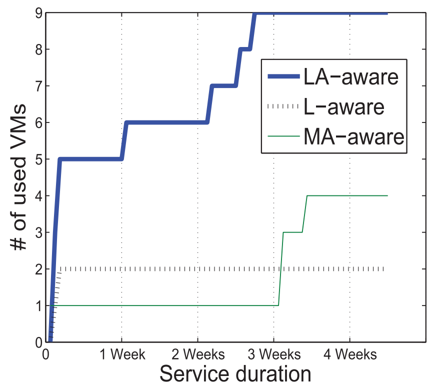


Fig. 16. # of used VMs with different service durations.

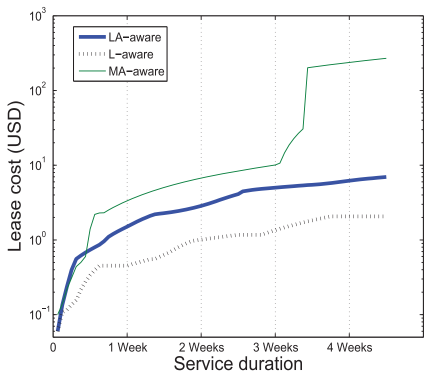


Fig. 17. Lease cost with different service durations.

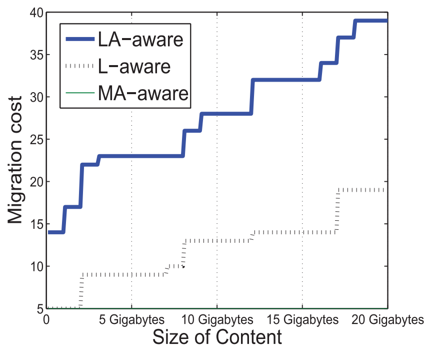


Fig. 18. Migration cost with different content sizes.

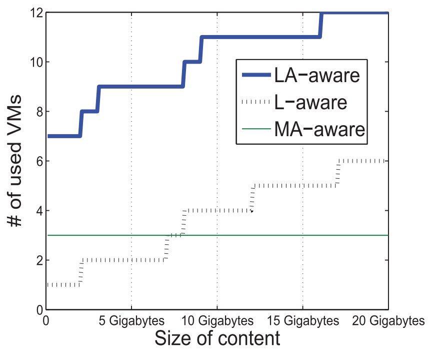


Fig. 19. # of used VMs with different content sizes.

close with the optimal results. Yet the complexity of LA-aware heuristic is much lower than the optimal algorithm, as discussed earlier.

Fig. 14b provides a further comparison when users want to deploy larger contents for one month (720 hours). This result shows that the lease cost is also quite sensitive with the increasing of contents size. This is because more VMs will be used to hold larger content for such a long service duration. It is worth noting that the service availability seems naturally increasing when more VMs are used in L-aware algorithm. However, this availability is depending on the randomly combined availability across all selected VMs, and there is no guarantee for the buyers.

LA-aware versus MA-aware. We now introduce the migration cost as well as the number of used VMs in our comparison. Fig. 15 shows the migration cost when buyers deploy their service from one week to more than four weeks. We can see that one drawback of the LA-aware algorithm is the increasing of migration cost. This is not surprising because the LA-aware algorithm is designed to schedule the contents across different VMs for lower lease cost. The MA-aware algorithm, on the other hand, can better reduce the migration cost for the buyers (with the migration cost less than 5 in the figure). As shown in Fig. 16, we can see that the MA-aware algorithm only used one VM to serve buyers' content for the first three weeks while the LA-aware algorithm used more than five VMs for lower lease cost.

As a tradeoff, minimizing the migration cost will naturally increase the lease cost. As shown in Fig. 17, we can see that the lease cost of MA-aware algorithm is the highest among all algorithms. In particular, its lease cost could be 10 times higher than that of the LA-aware algorithm. The L-aware algorithm, on the other hand, provides low lease cost and reasonable migration cost. However, it again suffers from the low service availability as shown in Fig. 21. Even worse, its service availability decreases very fast when the buyers want to deploy their service for longer durations.

Fig. 18 examines the migration cost when the buyers want to deploy larger contents for a fixed time duration (720 hours). We can see that the LA-aware algorithm will again give low lease cost but higher migration cost. The MA-aware algorithm, on the other hand, provides a constant migration cost of 0 using three VMs (Fig. 19). When we further check these three VMs, we find that they are all very stable VMs with high storage capacities. The selecting of these VMs can therefore give very low migration cost. However, as shown in Fig. 20, the lease cost of MA-aware algorithm is also quite high (around $200). This is almost 20 times higher than that of the LA-aware algorithm. It is worth noting that for a fixed service duration, selecting

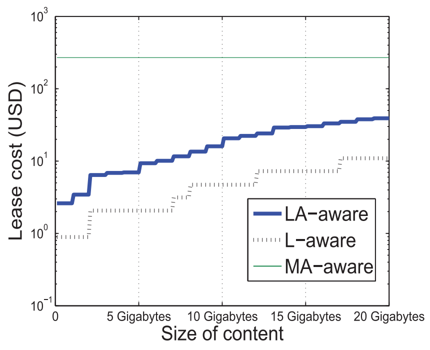


Fig. 20. Lease cost with different content sizes.

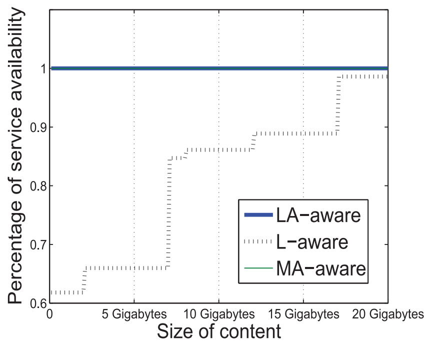


Fig. 22. Service availability with different content sizes.

more VMs will potentially increase the service availability. As shown in Fig. 22, the service availability of the L-aware algorithm is increasing with larger contents. Yet, as we have already discussed before, such a service availability cannot be guaranteed.

It is worth noting that our algorithms are flexible for buyers to set up a customized service availability, i.e., lower than 100 percent, for their applications. Fig. 23 shows the case when a buyer wants to deploy a 5 Gigabytes content for 720 hours with different service availabilities. Note that this figure refers only to the LA-aware algorithm. We can see that the buyers' service availability is linearly related with the lease cost. This figure clarifies the trade-off between service availability and lease cost. For example, the buyers will spend approximately \$0.6 to increase their service availability by 10 percent.

7 Conclusions and Further Discussions

This paper investigated the feasibility and the system design of enabling customer-provided resources for cloud computing. We closely examined the resource provisioning and pricing problems with dynamically available resources, and developed efficient solutions. We then presented the initial design of SpotCloud, a working system aiming at integrating the cloud resources from both enterprise and individual sellers. Trace analysis validated SpotCloud as a complement of great potentials to datacenter-based cloud.

Our work represents an initial attempt toward this direction, and there are many possible future avenues. Besides optimizing SpotCloud and exploring the best allocation of services between SpotCloud and datacenter-based cloud, we are particularly interested in the following four critical issues:

Privacy and security issues. It is worth noting that the utilization of customer-provided resources can introduce privacy and security issues to the cloud systems. Similar to the peer-assisted content storage systems, we believe that such a problem also needs to be carefully mitigated in our system. Not to mention the existing encryption functions in Spotcloud, one of our ongoing works is to design a privacy-aware load assignment for such a customer-provided cloud platform.

Random failure of customers. When a customer provides resources to SpotCloud, s/he needs to claim the periods that the local resources are available. The customers will also be motivated to well behave given the profit from providing resources. This is different from peer-to-peer networks where the peers can leave the system freely, and thus the online/offline behaviors are much more predictable in our system. In the rare case of uncontrollable random failures, some smart backup algorithms can be explored for fault recovery. In particular, we aim to quantify the similarity of VMs' online availability, optimize the periodical status reporting and organize them into a binary tree structure to backup each other. Applying Amazon *Elastic Block Store* (EBS) service with necessary revisions is also a possible option.

Migration cost with finer granularity. It is worth noting that the migration cost defined in this study only considers the frequency of migrations and serves as an approximation of the real migration cost. In practice, actual migration cost of different applications can be different even when their migration frequencies are identical; for example, the real

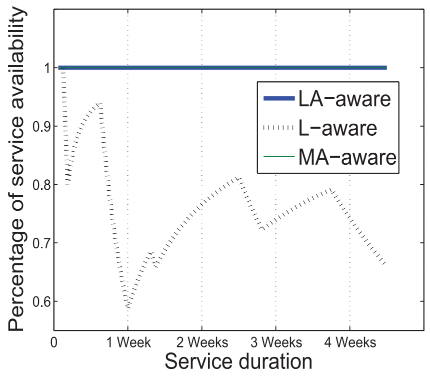


Fig. 21. Service availability with different service durations.

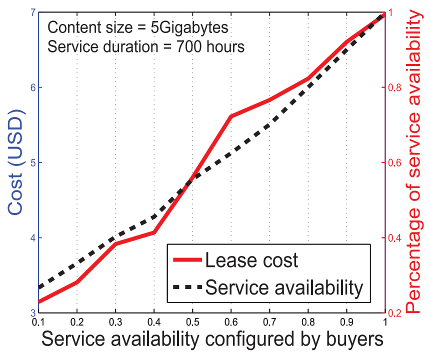


Fig. 23. Trade-off between cost and service availability.

migration cost can depend on the application protocol, the content size or the bandwidth between VMs. Therefore, this migration cost could be better defined given the detailed characteristics of different applications. Such information might also facilitate smart allocations between SpotCloud and datacenters.

Modularized system enhancement. It is known that high cost and high availability do not necessarily be better than low cost and low availability in real-world system deployments. Instead of considering everything for the users, it is also important to modularize different performance metrics and let the users decide their own priorities/tradeoffs. In our SpotCloud system, such an enhancement will be able to fulfill different user demands with lower costs.

ACKNOWLEDGMENTS

H. Wang's work was supported by Chancellor's Small Grant and a Start-up Grant from the University of Minnesota Duluth. J. Liu's work was supported by a Canada NSERC Discovery Grant, an NSERC Strategic Project Grant, and a China NSFC Major Program of International Cooperation Grant (61120106008). D. Wang's work was partially supported by Hong Kong PolyU/1-ZVC2 and G-UB72. A preliminary version of this paper appeared in Proc. INFOCOM'12.

REFERENCES

[1] M. Hajjat, X. Sun, Y. E. Sung, D. Maltz, S. Rao, K. Sripanidkulchai, and M. Tawarmalani, "Cloudward bound: Planning for beneficial migration of enterprise applications to the cloud," in *Proc. ACM SIGCOMM Conf.*, 2010, pp. 243–254.

[2] Amazon web service. (2014). [Online]. Available: http://aws.amazon.com/

[3] GoGrid Cloud Hostin. (2014). [Online]. Available: http://gogrid.com/

[4] Google AppEngine. (2014). [Online]. Available: http://code.google.com/appengine/

[5] Microsoft Windows Azure. (2014). [Online]. Available: http://www.microsoft.com/

[6] Rackspace Cloud. (2014). [Online]. Available: http://www.rackspacecloud.com/

[7] M. Armbrust, R. G. A. Fox, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," Univ. California, Berkeley, CA, USA, Tech. Rep., 2009.

[8] K. Sripanidkulchai, S. Sahu, Y. Ruan, A. Shaikh, and C. Dorai, "Are clouds ready for large distributed applications?" in *Proc. SOSP LADIS Workshop*, 2009, pp. 18–23.

[9] SETI@HOME. (2014). [Online]. Available: http://setiathome.berkeley.edu/

[10] S. Garfinkel, "An evaluation of amazon s grid computing services : EC2 , S3 and SQS," Harvard Univ., Cambridge, MA , USA, Tech. Rep., 2008.

[11] E. Walker, "Benchmarking amazon EC2 for high-performance scientific computing," *USENIX Login*, vol. 33, pp. 18–23, 2008.

[12] A. Li and X. Yang, "CloudCmp: Comparing public cloud providers," *Proc. ACM/USENIX Conf. Internet Meas.*, 2010, pp. 1–14.

[13] J. S. Ward, "A performance comparison of clouds: Amazon EC2 and ubuntu enterprise cloud," *Proc. SICSA DemoFEST*, pp. 1–14, 2009.

[14] T. Wood, P. Shenoy, and Arun,, "Black-box and gray-box strategies for virtual machine migration," in *Proc. USENIX Conf. Netw. Syst. Design Implementation*, 2007, p. 17.

[15] V. Shrivastava, P. Zerfos, K. Lee, H. Jamjoom, Y. Liu, and S. Banerjee, "Application-aware virtual machine migration in data centers," in *Proc. IEEE INFOCOM*, 2011, pp. 66–70.

[16] Y. Seung, T. Lam, L. E. Li, and T. Woo, "Seamless scaling of enterprise applications into the cloud," in *Proc. IEEE INFOCOM*, 2011, pp. 211–215.

[17] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "Kingfisher: Cost-aware elasticity in the cloud," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, 2011, pp. 206–210.

[18] Y. Wu, C. Wu, B. Li, X. Qiu, and F. Lau, "CloudMedia: When cloud on demand meets video on demand," in *Proc. IEEE 31st Int. Conf. Distrib. Comput. Syst.*, 2011, pp. 268–277.

[19] S. Kannan, A. Gavrilovska, and K. Schwan, "Cloud4Home—Enhancing data services with home clouds," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, 2011, pp. 539–548.

[20] A. Fox, S. D. Gribble, Y. Chawathe, E. A. Brewer, and P. Gauthier, "Cluster-based scalable network services," in *Proc. 16th ACM Symp. Oper. Syst. Principles*, 1997, pp. 78–91.

[21] J. Chase, D. Anderson, P. Thakar, A. Vahdat, and R. Doyle, "Managing energy and server resources in hosting centers," *ACM SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 103–116, 2001.

[22] L. Grit, D. Irwin, A. Yumerefendi, and J. Chase, "Virtual machine hosting for networked clusters: Building the foundations for autonomic orchestration," in *Proc. 2nd Int. Workshop Virtualization Technol. Distrib. Comput.*, 2006, p. 7.

[23] T. F. Abdelzaher, K. G. Shin, and N. Bhatti, "Guarantees for web server end-systems: A control-theoretical approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 80–96, Jan. 2002.

[24] D. Villela, P. Pradhan, and D. Rubenstein, "Provisioning servers in the application tier for e-commerce systems," in *Proc. IEEE 12th Int. Workshop Quality Service*, 2004, pp. 57–66.

[25] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Commun. Surv. Tutorials*, vol. 7, no. 2, pp. 72–93, Second Quarter 2005.

[26] Planetlab. (2014). [Online]. Available: http://www.planet-lab.org/

[27] Seller API and Third party provider integration guide. (2014). [Online]. Available: http://spotcloud.com/fileadmin/docs/SpotCloudProviderGuide.pdf

[28] AWS Management Console. (2014). [Online]. Available: http://aws.amazon.com/console/

[29] J. L. H. Wang, F. Wang, and J. Groen, "Measurement and utilization of customer-provided resources for cloud computing," *Proc. IEEE INFOCOM*, 2012, pp. 442–450.

[30] Amazon EC2 spot instances. (2014). [Online]. Available: http://aws.amazon.com/ec2/spot-instances/

[31] K. Zhu, "Information transparency of business-to-business electronic markets: A game-theoretic analysis," *Manage. Sci.*, vol. 50, no. 5, pp. 670–685 , 2004.

[32] M. Beckmann, "Bertrand-edgeworth duopoly revisited," in *Operations Research Verfahren III*, R. Henn, Ed. Meisenheim, Germany: Verlag Anton Hain, 1965, pp. 55–68.

[33] J. Farrell and E. Maskin, "Renegotiation in repeated games," *J. Econ. Theory*, vol. 1, no. 4, pp. 327–360, 1989.

[34] M. Kandori, "Introduction to repeated games with private monitoring," *J. Econo. Theory*, vol. 102, no. 1, p. 1–15, 2002.

[35] D. Abreu and A. Rubinstein, "The structure of nash equilibrium in repeated games with finite automata," *J. Econometric Soc.*, vol. 56, no. 6, pp. 1259–1281, 1988.

[36] K. Krauter, R. Buyya, and M. Maheswaran, "A taxonomy and survey of grid resource management systems for distributed computing," *Softw.: Practice Exp.*, vol. 32, no. 2, pp. 135–164, 2002.

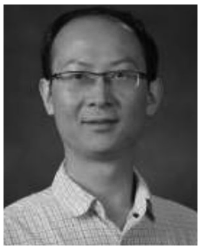
[37] J. Liu, S. G. Rao, B. Li, and H. Zhang, "Opportunities and challenges of peer-to-peer internet video broadcast," *Special Issue Recent Adv. Distrib. Multimedia Commun.*, vol. 96, no. 1, pp. 11–24, 2008.

[38] A. W. Kolen, J. K. Lenstra, C. H. Papadimitriou, and F. C. Spieksma4, "Interval scheduling: A survey," *Naval Res. Logistics*, vol. 54, no. 5, pp. 530–543, 2007.

[39] V. Paxson and S. Floyd, "Wide-area traffic: The failure of Poisson modeling," *Proc. ACM SIGCOMM*, pp. 226–244, 1994.

[40] H. Wang, F. Wang, and J. Liu. (2011). Measurement and gaming analysis of SpotCloud. Simon Fraser Univ., Burnaby, BC, Canada, Tech. Rep. [Online]. Available: http://netsg.cs.sfu.ca/spdata/sc.pdf.

Haiyang Wang received the PhD degree in computing science from Simon Fraser University, Burnaby, BC, Canada, in 2013. He is currently an assistant professor at the Department of Computer Science, University of Minnesota Duluth, MN. His research interests include cloud computing, big data, socialized content sharing, multimedia communications, and peer-to-peer networks. He is a member of the IEEE.



Feng Wang (S'07-M'13) received both the bachelor's and master's degrees in computer science and technology from Tsinghua University, Beijing, China, in 2002 and 2005, respectively. He received the PhD degree in computing science from Simon Fraser University, Burnaby, BC, Canada in 2012. He received the Chinese Government Scholarship for Outstanding Self-financed Students Studying Abroad (2009) and IEEE ICME Quality Reviewer Award (2011). He is currently an assistant professor at the Department of Computer and Information Science, University of Mississippi, University, MS. His research interests include cloud computing, wireless mesh/sensor networks, cyberphysical systems, peer-to-peer networks, and socialized content sharing. He is a TPC cochair in IEEE International Conference on Communications in China (ICCC) 2014 for Symposium on Wireless Networking and Multimedia. He also serves as a TPC member in various international conferences such as IEEE/ACM IWQoS, ACM Multimedia, IEEE ICC, IEEE GLOBECOM, IEEE CloudCom, and IEEE ICME. He is a member of the IEEE.

J. Liu received the BEng degree (cum laude) from Tsinghua University, Beijing, China, in 1999, and the PhD degree from The Hong Kong University of Science and Technology in 2003, both in computer science. He is a co-recipient of ACM TOMCCAP Nicolas D. Georganas Best Paper Award 2013, ACM Multimedia Best Paper Award 2012, IEEE Globecom 2011 Best Paper Award, and IEEE Communications Society Best Paper Award on Multimedia Communications 2009. He is a Full Professor in the School of Computing Science, Simon Fraser University, British Columbia, Canada, and an EMC-Endowed Visiting Chair Professor of Tsinghua University, Beijing, China. He was an Assistant Professor in the Department of Computer Science and Engineering at The Chinese University of Hong Kong from 2003 to 2004. His research interests include multimedia systems and networks, cloud computing, social networking, wireless ad hoc and sensor networks, and peer-to-peer and overlay networks. He serves on the editorial boards of IEEE Transactions on Multimedia, IEEE Communications Surveys and Tutorials, IEEE Access, and IEEE Internet of Things Journal. He is a TPC co-chair of IEEE/ACM IWQoS'14 and an Area Chair of ACM Multimedia'14.

Dan Wang received the BSc degree from Peking University, Beijing, the MSc degree from Case Western Reserve University, Cleveland, OH, and the PhD degree from Simon Fraser University, Vancouver, Canada, all in computer science. He is an associate professor at the Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong. His research interest includes sensor networks, internet routing, and applications. He is a senior member of the IEEE.