On Resource Allocation and Pricing in Hybrid Cloud: A Hierarchical Approach

***Abstract*—Online service providers need to deal with heterogeneous and time-variant user requests, and they often utilize cloud computing technologies to cope with this diversity. In this paper, we propose a hybrid cloud approach, where routine requests are normally addressed by local private cloud, while a third-party public cloud is needed when there is a burst of incoming requests. Our goal is to answer 1) from the online service provider's perspective, how to decide the local private cloud resource allocation, and how to distribute the incoming requests to private and/or public clouds; and 2) from the public cloud provider's perspective, how to decide the optimal prices to sell these public cloud resources so as to maximize its profit. We use a Stackelberg game model to capture the complex interactions between users, online service providers and public cloud providers, based on which we build a hierarchical framework to analyze resource allocation and pricing strategies from static to dynamic cases. We design efficient online algorithms to determine the public cloud provider's and the online service provider's optimal decisions. Via learning method we extract the features of user requests and use real-trace simulations to validate the effectiveness of our design.**

I. Introduction

Recently online services are applied into many aspects of our daily life. For example, people perform communication, shopping, entertainment and professional activities over the Internet. The Online Service Providers (OSPs) need to develop new technologies, aiming at providing QoS guaranteed service while at the same time, minimizing its operating cost. Often times, an OSP operates a number of distinct services simultaneously, so it needs to answer requests from various applications that differ a lot in pattern. To this end, many OSPs develop private cloud systems so as to provide virtualized environment where each service can be properly addressed.

However, private cloud is still not enough. Users' requests can be of high variance over time. In general, we can roughly classify the customers' request patterns into two categories:

- Routine case: Normal and periodical user requests, such as webpage browsing or video viewing, can be roughly estimated over the time span according to their regularity.
- Burst case: Sudden increase in request amount, for example, the Black Friday online shopping, can incur a much higher burden, e.g., hundreds of times higher, than the routine case.

A naive way for an OSP to cope with these requests is to prepare a very powerful private cloud system, such that when the burst occurs, it can still answer all requests. However, this approach is neither economically efficient nor robust. The hardware and management cost is surely very high, and for most of the time, the request amount is much lower than the peak time, so many servers remain idle. Even if one prepares such a system, it can hardly be guaranteed to work normally at the peak time. In fact, there have been many cases that online shopping sites crashed in shopping festivals (e.g., the Black Friday). Researchers may also remember that during the last few hours before the submission deadline of *INFOCOM* 2013, the EDAS system crashed.

Alternatively, hybrid cloud is a promising solution to this problem. Hybrid cloud means a combination of a private cloud operated by the OSP, and a public cloud system operated by a third-party Cloud Service Provider (CSP). To address the routine case, most requests can be answered by the local private cloud; and for the burst case, the OSP can resort those requests beyond its capacity to the public cloud.

The fundamental questions of the hybrid cloud approach are resource allocation and pricing. In particular, given the request pattern of users, from the OSP's perspective, it has to decide how to build up a private cloud in terms of the amount of each resource (e.g., CPU, memory, storage etc.) to be prepared; and facing the incoming requests, how to properly distribute them into private/public clouds. From the CSP's perspective, it needs to decide how to set prices to sell cloud services so as to maximize its profit.

These questions are non-trivial to answer due to the following challenges. First, the stochastic nature of the request arrival pattern makes it difficult to determine the local resource arrangement. Second, since the arrival rate of requests is very high in a large scale system, the resource allocation algorithm must be dynamic and time efficient, but finding the optimal allocation is a very complex problem: even if we know the request distribution a priori, it is still NP-complete. Last but not least, the CSP's pricing decision and the OSP's allocation and purchasing decisions are highly coupled, so we need to deeply understand their economic correlations and dynamics. All these challenges make the resource allocation and pricing problem in hybrid cloud an open question so far.

To tackle these challenges, we propose a hierarchical approach to analyze the hybrid cloud framework. In particular, we use a Stackelberg game to capture the interactions between the OSP and the CSP, based on which we design algorithms to decide the OSP's strategy in purchasing the public cloud resources, as well as its allocation mechanism for the incoming requests. We also design an algorithm for the CSP to decide its pricing scheme that approximately maximizes its profit. Due to the NP-completeness of each sub-problem, we design efficient and effective approximation algorithms, and use theoretic analysis and simulations to validate their high accuracy.

TABLE I. Main Notations

Notation	Explanation
$\mathcal{N}$	resources in the private cloud
$\mathcal{R}$	set of request types
$\vec{\varphi}_t$	instantaneous amount of incoming requests
$\mathcal{P}_l$	OSP's unit operating cost
$\mathcal{P}_c$	price for public cloud resources
$\mathbf{I}_t$	set of users' requests
$\mathbf{L}_t$	subset of requests allocated to OSP
$\mathbf{C}_t$	subset of requests allocated to CSP
$\mathcal{S}_I^t$	resource demand of $\mathbf{I}_t$
$\mathcal{S}_L^t$	resource demand of $\mathbf{L}_t$
$\mathcal{S}_C^t$	resource demand of $\mathbf{C}_t$

To sum up, we design a comprehensive framework to design resource allocation and pricing in hybrid cloud. Our contributions are:

- We reveal the interactions of the OSP and the CSP, and present a hierarchical analytical framework for the hybrid cloud.
- We propose approximation algorithms for resource allocation and pricing, and show their efficiency and accuracy both theoretically and empirically.
- Base on real-trace simulation, we show our design can achieve satisfactory performance in practice.

This is the outline of this paper. In Section II, we formulate the hybrid cloud as a Stackelberg game model. Section III proposes two approximation algorithms to address the static request distribution problem for the OSP. Based on this, we derive the dynamic resource allocation algorithm for the OSP in Section IV. In Section V we use an approximation algorithm to decide the optimal pricing scheme for the CSP. Real-trace evaluation is given in Section VI. Section VII states related work and Section VIII concludes. Due to page limit, some of our detailed derivations are omitted in this paper; interested readers may refer to our technical report [1] for details.

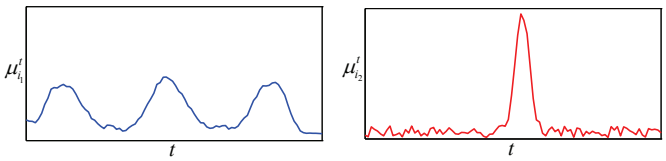
II. System and Game Model

In this section, we use mathematical models to capture the hybrid cloud system applied in online services. Based on this model, we will develop a game theoretic model to capture the interactions between the OSP and the CSP.

A. Problem Formulation

We assume that users' requests can be classified into n types. In reality, they can represent various applications. Each request type requires a specific collection of different computing resources, for example, CPU, memory and storage. We can define $\mathcal{R}$ as the set of all request types, i.e.,

$$\mathcal{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_i, \ldots, \mathcal{R}_n\}, i = 1, 2, \ldots, n, \quad (1)$$



(a) a routine case (b) a burst case

Fig. 1. Examples of μ_i^t

where $\mathcal{R}_i = (r_{i1}, \ldots, r_{ij}, \ldots, r_{im})$ is a type i request. Each element r_{ij} denotes the amount of resource j required to satisfy $\mathcal{R}_i$.

Assume that the number of incoming request in type i, i.e., $\mathcal{R}_i$, is a random variable K_i in any given time t. Without loss of generality, we formulate K_i to be an independent Gaussian distribution as

$$K_i^t : \mathcal{G}_i^t(k_i) = \frac{1}{\sqrt{2\pi}\sigma_i^t} \exp\left[-\frac{(k_i - \mu_i^t)^2}{2(\sigma_i^t)^2}\right] \sim N(\mu_i^t, \sigma_i^t), \quad (2)$$

where the parameter μ_i^t and σ_i^t are given distributions over t, and they represent the request incoming patterns. Figure 1 shows μ_i^t in a routine or burst case, respectively. We use $\vec{\varphi}_t$ to represent the instantaneous amount of incoming requests in all types, which can be formulated as

$$\vec{\varphi}_t = \left(K_1^t, \ldots, K_i^t, \ldots, K_n^t\right). \quad (3)$$

Now let us consider how to capture the property of the private cloud system operated by the OSP. Let $\mathcal{N}$ be the amount of resources possessed by this private cloud, then it can be represented by

$$\mathcal{N} = (\mathcal{N}_1, \ldots, \mathcal{N}_j, \ldots, \mathcal{N}_m), j = 1, 2, \ldots, m, \quad (4)$$

where each element represents the amount of each resource available in the private cloud, e.g., CPU, memory, storage, etc. For each kind of resource, it is associated with a unit operating cost, denoted by $\mathcal{P}_l = (p_{l1}, \ldots, p_{lj}, \ldots, p_{lm})$.

When requests cannot be fully satisfied in the private cloud, they are resorted to the public cloud. A classic methodology, Vector Bin Packing [2] [3] provides an optimal combination of Virtual Machines (VMs) to cover any given set of request demands. On measuring the fee paid by the OSP to the CSP, we can view it as the sum of fee paid for each kind of resource, where the unit price vector is $\mathcal{P}_c = (p_{c1}, \ldots, p_{cj}, \ldots, p_{cm})$.

Now let us focus on the profit of the OSP and the CSP at a particular time point t. Let $\mathbf{I}_t$ be the set of requests from users at time t. The OSP allocates a subset of these requests $\mathbf{L}_t$ to its private cloud, and the rest $\mathbf{C}_t$ to the public cloud. Resource demand of $\mathbf{I}_t$, $\mathbf{L}_t$ and $\mathbf{C}_t$ are denoted as $\mathcal{S}_I^t$, $\mathcal{S}_L^t$ and $\mathcal{S}_C^t$, respectively, each demand being a vector of resources needed to satisfy the corresponding requests. Obviously, we have $\mathcal{S}_I^t = \mathcal{S}_L^t + \mathcal{S}_C^t$. Upon satisfying a request, the OSP owns a profit for this service provided. We denote $\mathcal{U}_t$ as the sum of this utility at time t. We can express the utility (or payoff[1]) of

[1]In this paper, we use "utility" and "payoff" interchangeably.

the OSP and the CSP as

$$\Pi_L^t = \mathcal{U}_t - \mathcal{N}\mathcal{P}_l - \mathcal{S}_C^t\mathcal{P}_c = \Pi_L^t\left(\mathcal{P}_c, \mathcal{N}\right), \tag{5}$$

$$\Pi_C^t = \mathcal{S}_C^t\mathcal{P}_c = \Pi_C^t\left(\mathcal{P}_c, \mathcal{N}\right), \tag{6}$$

where the OSP's utility equals the utility of satisfying all requests minus its cost, and the CSP's utility equals its revenue.

Tabel I lists important notations introduced in this section.

B. A Stackelberg Game Model

In this paper, we use a Stackelberg game [4] to capture the interactions between the OSP and the CSP, where the CSP is the leader and the OSP is the follower. We use this hierarchical game model because in reality, the CSP decides its public cloud resource sale scheme first. Once the decision is made, the CSP usually does not change it frequently. Based on this scheme, the OSP decides its local resource preparation accordingly in order to maximize its own utility. We formally describe the game as follows.

- **Players.** The online service provider (OSP) and the cloud service provider (CSP).
- **Payoff.** The payoff of the OSP is

$$\Pi_L\left(\mathcal{P}_c, \mathcal{N}\right) = \int_t \Pi_L^t\left(\mathcal{P}_c, \mathcal{N}\right) \mathrm{d}t, \tag{7}$$

  and the payoff of the CSP is

$$\Pi_C\left(\mathcal{P}_c, \mathcal{N}\right) = \int_t \Pi_C^t\left(\mathcal{P}_c, \mathcal{N}\right) \mathrm{d}t. \tag{8}$$

  We will formally describe these payoff functions in later sections.
- **Game strategy.** The CSP decides the pricing strategy $\mathcal{P}_c$ for the public cloud. The OSP decides the resource $\mathcal{N}$ powered by the private cloud.

The solution to this game is called Stackelberg equilibrium (or SE), which can be solved by applying the backward induction [4] as follows.

1) Assume the CSP fixes $\mathcal{P}_c$. The OSP solves the problem $\mathcal{N}^*\left(\mathcal{P}_c\right) = \arg\max_{\mathcal{N}} \Pi_L\left(\mathcal{P}_c, \mathcal{N}\right)$.
2) By knowing $\mathcal{N}^*\left(\mathcal{P}_c\right)$, the CSP solves $\mathcal{P}_c^* = \arg\max_{\mathcal{P}_c} \Pi_C\left(\mathcal{P}_c, \mathcal{N}^*\left(\mathcal{P}_c\right)\right)$.

Our following formulation will show that both payoff functions are continuous with respect to the decision variables, thus the two steps are both optimization problems over a compact set, so the existence of SE is guaranteed.

Correspondingly, the next sections are organized in a backward manner. In Section III, we decide $\mathbf{L}_t$ for any given $\mathcal{N}$ and $\mathcal{P}_c$. This serves as the basis of our game analysis. Section IV analyzes the OSP's problem for any given CSP's strategy. Section V decides the CSP's optimal pricing scheme so as to obtain the Stackelberg equilibrium.

III. Static Request Distribution: Private or Public

Priori to solving the Stackelberg game, let us first consider given the strategies of the OSP and the CSP, how can the OSP allocate the incoming requests into the private and public clouds so as to maximize its utility. In this section, we formulate this problem into an optimization, prove its NP-completeness, and then present two approximation algorithms for efficient solutions.

A. The Static Optimization Problem

For given private cloud resource $\mathcal{N}$, an OSP needs to decide which requests to be allocated in the private cloud for processing, and which are resorted to the public cloud. Note that due to the various nature of requests, they have different "appropriateness" to be put in local. For example, requests with sensitive information, high security requirement or high data transmission cost should be processed locally with a high priority. In here, we use a notation λ_{ip} to represent this appropriateness of putting the p^{th} request of the i^{th} type in the private cloud. Later we use Λ_t to represent the matrix consisting of elements λ_{ip}. We use a binary variable x_{ip} to denote whether this request is allocated in the private cloud, i.e., $x_{ip} = 1$ if yes or 0 otherwise. In general, the OSP aims at solving the following problem, denoted by OPT:

$$\begin{aligned} \max \quad & \sum_{i=1}^{n}\sum_{p=1}^{K_i^t} \lambda_{ip} x_{ip} \\ \text{subject to} \quad & \sum_{i=1}^{n}\sum_{p=1}^{K_i^t} r_{ij} x_{ip} \leqslant \mathcal{N}_j, \forall 1 \leq j \leq m, \\ & x_{ip} \in \{0,1\}. \end{aligned} \tag{9}$$

In this subsection, we treat OPT as a static optimization at t, and based on this we will further determine the OSP's best choice in the next section. Now let us prove that OPT is NP-complete.

Theorem 1. *OPT is NP-complete.*

Proof. Given any solution to OPT, we can verify the solution in a polynomial time. Thus OPT is NP.

Then we build OPT from the Binary Knapsack Problem (BKP), one of Karp's 21 NP-complete problems [5] [6]. The decision of BKP is NP-complete and its optimization is NP-hard. A dimensional extension of BKP leads to the Multiple Knapsack Problem (MKP), which is at least NP-hard [7] [8]. Suppose we have a knapsack with a limit of $\{\mathcal{N}_1, \ldots, \mathcal{N}_j, \ldots, \mathcal{N}_m\}$. Given a set of items, each with a weight of $(r_{i1}, \ldots, r_{ij}, \ldots, r_{im})$ and a value of λ_i. Next we determine an item collection to maximize its value under the constraint of the sum of weights. Therefore, OPT reduces to MKP and is thus NP-complete. □

Given that OPT is NP-complete, we design approximation algorithms to achieve the sub-optimality in an efficient manner. In particular, we propose the greedy mechanism and the load balancing mechanism. The first one has an advantage in the low computational complexity, while the latter is nearer to the optimality.

Algorithm 1: $APPGS$ Mechanism

Input: $\vec{\varphi}_t$, Λ_t, $\mathcal{N}$
Output: private set $\mathbf{L}_t$, public set $\mathbf{C}_t$

Initialization:
$V = \sum_{i=1}^{n} K_i^t$, $m_v = \{r_{v1}, r_{v2}, \ldots, r_{vm}\}$ $(v = 1, 2, \ldots, V)$, $\mathbf{L}_t = \emptyset$, $\mathbf{C}_t = \emptyset$, $\mathbf{T} = \emptyset$
Sorting request list $\{m_1^*, m_2^*, \ldots, m_V^*\}$ such that $\frac{\lambda_1^*}{\sqrt{\sum_{j=1}^{m} r_{1j}^2}} \geqslant \frac{\lambda_2^*}{\sqrt{\sum_{j=1}^{m} r_{2j}^2}} \geqslant \ldots \geqslant \frac{\lambda_V^*}{\sqrt{\sum_{j=1}^{m} r_{Vj}^2}}$
$\mathbf{I} = \{m_1^*, m_2^*, \ldots, m_V^*\}$
for $v = 1 : V$ **do**
  if $\mathbf{T} + m_v^* \leqslant \mathcal{N}$ **then**
    $\mathbf{L}_t := \mathbf{L}_t \bigcup \{m_v^*\}$
    $\mathbf{T} := \mathbf{T} + m_v^*$
  end
end
Return $\mathbf{L}_t, \mathbf{C}_t = \mathbf{I} - \mathbf{L}_t$

B. A Greedy Sorting Mechanism for OPT

Now we design the following Approximated Greedy Sorting ($APPGS$) algorithm depicted in Algorithm 1. Let $\{\mathbf{L}_t, \mathbf{C}_t\} = APPGS\left(\vec{\varphi}_t, \Lambda_t, \mathcal{N}\right)$ be the output of this algorithm. The algorithm consists of three steps:

- Step 1: Initialization. In this step, all user requests are randomly arranged and denoted by the term m_v.
- Step 2: Sorting. We sorts the requests in a sequence such that $\frac{\lambda_v^*}{\sqrt{\sum_{j=1}^{m} v_{vj}^2}}$ is non-decreasing in v. We use $\mathbf{I}$ to denote this sequence and m_v^* to denote the index of each request.
- Step 3: Allocation. We use the greedy algorithm to generate $\mathbf{L}_t$, the set of requests to process in the private cloud, and also $\mathbf{C}_t$, the set of requests to the public cloud, as $\mathbf{I} - \mathbf{L}_t$.

We next analyze the time complexity of Algorithm 1.

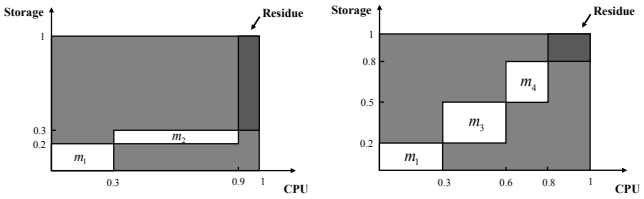
Theorem 2. *The time complexity of Algorithm 1 is polynomial.*

Proof. Let the number of user requests be N. In Step 2, the sorting process is of $O\left(N \log N\right)$ complexity. In Step 3, we need to traverse N user requests, and its computational complexity is $O\left(N\right)$. In conclusion, the computational complexity of Algorithm 1 is $O\left(N \log N\right)$. □

C. A Load Balancing Mechanism for OPT

In this subsection, we present another approximation algorithm with a better utility performance but a bit higher complexity. We call it Approximated Load Balancing ($APPLB$) algorithm [9].

Let us use a toy example to illustrate the intuition of our algorithm. We consider requests associated with two dimensional resource requirements, i.e., CPU and storage. In Figure 2(a), the private cloud accepts requests m_1 and m_2, but the rest available resources are not enough for any other incoming requests. But if the OSP chooses m_1, m_3 and m_4 into the private cloud, its resource utilization ratio is higher



(a) unbalancing load (b) balancing load

Fig. 2. Comparison of load decision

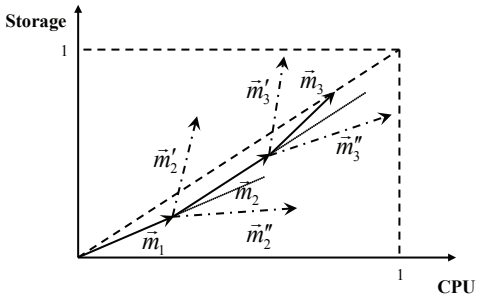


Fig. 3. A Simple Diagram of Vector Selection

(see Figure 2(b)). Thus, our objective is to design an algorithm that maximizes the resource utilization in the private cloud.

Now we derive the request selection for m-dimensional resource in a vector view. Each request can be denoted as

$$\begin{aligned} \vec{m}_v &= \alpha_v^1 \vec{e}_1 + \ldots + \alpha_v^j \vec{e}_j + \ldots + \alpha_v^m \vec{e}_m \\ &= \left(\alpha_v^1, \ldots, \alpha_v^j, \ldots, \alpha_v^m\right), \ v = 1, 2, \ldots, V \end{aligned} \tag{10}$$

where $\vec{e}_j$ is the unit vector of the j^{th} resource type. When selecting the next request denoted by $\vec{m}_{v+1}$, we minimize the dot product $\langle \vec{m}_v, \vec{m}_{v+1} \rangle$. The intuition of our approach is shown in Figure 3, where we select the vector $\vec{m}_2$ by minimizing the dot product $\langle \vec{m}_1, \vec{m}_2 \rangle$. Selection of $\vec{m}_3$ also follows the same rule. Now let us depict the load balancing algorithm $\{\mathbf{L}_t, \mathbf{C}_t\} = APPLB\left(\vec{\varphi}_t, \Lambda_t, \mathcal{N}\right)$.

In each outer for-loop, Algorithm 2 runs as follows:

- Step 1: Find requests who can be addressed by private cloud into the set $\mathbf{T}$.
- Step 2: We solve $\arg\min_{\vec{m}_\theta \in \mathbf{T}} \langle \vec{m}_v, \vec{m}_\theta \rangle$ and place the result into $\mathbf{H}$. If more than one solution exist, choose the one with the largest appropriateness. Denote our choice as $\vec{\theta}^*$.
- Step 3: Store $\vec{\theta}^*$ into $\mathbf{L}_t$, and update $\mathcal{N}$, $\vec{m}_{v+1}$, $\mathbf{T}$ and $\mathbf{I}_t$ for the next loop.

Comparing with Algorithm 1, or $APPGS$, the $APPLB$ algorithm is more efficient in terms of the OSP's payoff. Although a bit more complex than $APPGS$, Theorem 3 shows that $APPLB$ is still efficient in time consumption.

Theorem 3. *The time complexity of Algorithm 2 is polynomial.*

Proof. Assume the OSP has received N user requests. At the v^*-th loop round, we need to find the candidates who fit the current resource constraint, i.e., traversing $N - v^*$ residual requests. Until the termination time, the algorithm runs for

Algorithm 2: $APPLB$ Mechanism

Input: $\vec{\varphi}_t$, Λ_t, $\mathcal{N}$
Output: private set $\mathbf{L}_t$, public set $\mathbf{C}_t$

Initialization:
$V = \sum_{i=1}^{n} K_i^t$, $\vec{m}_v = r_{v1}\vec{e}_1 + \ldots + r_{vm}\vec{e}_m$ $(v = 1, 2, \ldots, V)$, $\mathbf{L}_t = \emptyset$, $\mathbf{C}_t = \emptyset$, $\mathbf{T} = \emptyset$, $\mathbf{H} = \emptyset$, $\mathbf{I}_t = \{\vec{m}_1, \ldots, \vec{m}_V\}$, $\vec{m}_0 = \mathbf{0}$
for $v = 0 : V - 1$ **do**
  for $\vec{m}_j \in \mathbf{I}$ **do**
    if $\mathcal{N} - \vec{m}_j \geqslant \mathbf{0}$ **then**
      $\mathbf{T} = \mathbf{T} \bigcup \{\vec{m}_j\}$
    end
  end
  $\mathbf{H} = \arg\min_{\vec{m}_\theta \in \mathbf{T}} \langle \vec{m}_v, \vec{m}_\theta \rangle$
  $\vec{\theta}^* = \arg\max_{\vec{\theta} \in \mathbf{H}} \lambda\left(\vec{\theta}\right)$
  $\mathbf{L}_t := \mathbf{L}_t \bigcup \left\{\vec{\theta}^*\right\}$, $\mathcal{N} := \mathcal{N} - \vec{\theta}^*$
  $\vec{m}_{v+1} := \vec{m}_v + \vec{\theta}^*$, $\mathbf{T} := \emptyset$
  $\mathbf{I} := \mathbf{I}_t - \left\{\vec{\theta}^*\right\}$
end
Return $\mathbf{L}_t, \mathbf{C}_t = \mathbf{I}_t - \mathbf{L}_t$

$N + (N-1) + \ldots + 1 = \frac{N^2+N}{2}$ times, where each time is only a dot product computing. In conclusion, the $APPLB$ has a polynomial computational complexity of $O\left(N^2\right)$. □

To summarize the two algorithms, $APPGS$ has a lower computational complexity while $APPLB$ is nearer to the optimality. In the remainder of this paper, we use them as a combination, called APP: when the computational size is not very large, we choose $APPLB$ to obtain a higher utility, but if the amount of requests is very large, we opt to apply $APPGS$ instead for better time efficiency. In later sections, we will further use simulations to validate the accuracy as well as efficiency of the two algorithms.

IV. OSP's Decision on Constructing a Private Cloud

Recall that in the previous section, we have established a Stackelberg game model which requires a backward induction to seek its solution. In this section, based on the results in the previous section, we solve the second stage of this game, i.e., the OSP's decision on constructing the private cloud. In short, we will solve $\mathcal{N}^*\left(\mathcal{P}_c\right) = \arg\max_{\mathcal{N}} \Pi_L\left(\mathcal{N}, \mathcal{P}_c\right)$.

A. Decision of Private Cloud

Considering the APP algorithm in Section III, we can rapidly compute the private set $\mathbf{L}_t$ and the public set $\mathbf{C}_t$ as

$$\{\mathbf{L}_t, \mathbf{C}_t\} = APP\left(\vec{\varphi}_t, \Lambda_t, \mathcal{N}\right). \tag{11}$$

Since $\mathbf{I}_t = \mathbf{L}_t \bigcup \mathbf{C}_t$, and $\mathcal{S}_I^t = \mathcal{S}_L^t + \mathcal{S}_C^t$, we can simplify Equation (11) to be

$$\mathcal{S}_L^t = APP\left(\vec{\varphi}_t, \Lambda_t, \mathcal{N}\right) = APP\left(\mathcal{N}, \vec{\varphi}_t\right). \tag{12}$$

Combining Equation (5) and (12), we obtain the OSP's utility as

$$\begin{aligned}\Pi_L^t\left(\mathcal{P}_c, \mathcal{N}\right) &= \mathcal{U}_t - \mathcal{N}\mathcal{P}_l - \mathcal{S}_C^t\mathcal{P}_c \\ &= \mathcal{U}_t - \left\{\mathcal{N}\mathcal{P}_l + \left[\mathcal{S}_I^t - APP\left(\mathcal{N}, \vec{\varphi}_t\right)\right]\mathcal{P}_c\right\}.\end{aligned} \tag{13}$$

In Equation (13), the terms $\mathcal{U}_t$, $\vec{\varphi}_t$, $\mathcal{P}_l$ and $\mathcal{P}_c$ are constant and do not change over time t. Therefore, the OSP's utility is a function of $\mathcal{N}$ for any given time t. For the OSP, the resources it need to prepare in the private cloud is determined by

$$\mathcal{N}^* = \arg\max_{\mathcal{N}} \int_t \Pi_L^t\left(\mathcal{P}_c, \mathcal{N}\right) \mathrm{d}t. \tag{14}$$

Since $\mathcal{U}_t$ and $\mathcal{S}_I^t\mathcal{P}_c$ are constants, we have

$$\begin{aligned}\mathcal{N}^* &= \arg\max_{\mathcal{N}} \int_t APP\left(\mathcal{N}, \vec{\varphi}_t\right)\mathcal{P}_c - \mathcal{N}\mathcal{P}_l \mathrm{d}t \\ &= \arg\max_{\mathcal{N}} \int_{t\in[0,T]} \int_{\vec{\varphi}_t \in \Theta} APP\left(\mathcal{N}, \vec{\varphi}_t\right)\mathcal{P}_c - \mathcal{N}\mathcal{P}_l \mathrm{d}\vec{\varphi}_t \mathrm{d}t \\ &= \arg\max_{\mathcal{N}} \ \mathcal{F}\left(\mathcal{N}, \mathcal{P}_c\right),\end{aligned} \tag{15}$$

where $\mathcal{F}\left(\mathcal{N}, \mathcal{P}_c\right) = \int_{t\in[0,T]} \int_{\vec{\varphi}_t \in \Theta} APP\left(\mathcal{N}, \vec{\varphi}_t\right)\mathcal{P}_c - \mathcal{N}\mathcal{P}_l \mathrm{d}\vec{\varphi}_t \mathrm{d}t$, $\Theta = \left(\mathcal{G}_1^t\left(k_1\right), \mathcal{G}_2^t\left(k_2\right), \ldots, \mathcal{G}_n^t\left(k_n\right)\right)$. In order to simplify the calculation, we proceed to show

$$\begin{aligned}\mathcal{F} &= \sum_{j=1}^{m} \int_{t\in[0,T]} \int_{\mathcal{S}_j^t} APP_j\left(\mathcal{N}_j, \mathcal{S}_j^t\right) p_{cj} - \mathcal{N}_j p_{lj} \mathrm{d}\mathcal{S}_j^t \mathrm{d}t \\ &= \sum_{j=1}^{m} \mathcal{F}_j\left(\mathcal{N}_j, p_{cj}\right),\end{aligned} \tag{16}$$

where $\mathcal{S}_j^t$ denotes the j^{th} resource demand at time t, and obviously, we have

$$\begin{aligned}\mathcal{S}_j^t : \mathcal{G}_j\left(s_j^t\right) &= \frac{1}{\sqrt{2\pi}\hat{\sigma}_j^t} \exp\left[-\frac{\left(s_j^t - \hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}\right] \\ &\sim N\left(\hat{\mu}_j^t, \hat{\sigma}_j^t\right),\end{aligned} \tag{17}$$

where the parameter $\hat{\mu}_j = \sum_{i=1}^{n} r_{ij}\mu_i^t$, $\hat{\sigma}_j = \sqrt{\sum_{i=1}^{n} r_{ij}^2\left(\sigma_i^t\right)^2}$.

If $\mathcal{S}_j^t < \mathcal{N}_j$, then the OSP can fully address the resource demand without purchasing public cloud resources. When $\mathcal{S}_j^t \geqslant \mathcal{N}_j$, the private cloud is fully utilized but still cannot satisfy all demands, so public cloud is needed. We can simply define $APP_j\left(\mathcal{N}_j, \mathcal{S}_j^t\right)$ as

$$APP_j\left(\mathcal{N}_j, \mathcal{S}_j^t\right) = \begin{cases} \mathcal{S}_j^t & \sum_i^n k_{imin} \leqslant \mathcal{S}_j^t < \mathcal{N}_j, \\ \mathcal{N}_j & \mathcal{N}_j \leqslant \mathcal{S}_j^t \leqslant \sum_i^n k_{imax}, \end{cases} \tag{18}$$

where k_{imim} (k_{imax}) denotes the minimal (maximal) number of requests of type i.

We assume that $\hat{\mu}_j$ and $\hat{\sigma}_j$ have certain distributions over time t, but we do not specify any particular form for them. We can still estimate the lower bound of $\mathcal{F}_j\left(\mathcal{N}_j, p_{cj}\right)$. Combining

Equation (16), (17) and (18), we have

$$\begin{aligned}\inf \mathcal{F}_j\left(\mathcal{N}_j, p_{cj}\right) = & \frac{p_{cj}\hat{\sigma}_j^t}{\sqrt{2\pi}} \int_0^T \rho\left(\mathcal{N}_j, t\right) \mathrm{d}t \\ & - \frac{p_{cj}+2p_{lj}}{2}\mathcal{N}_j T - \frac{p_{cj}}{2}\int_0^T \hat{\mu}_j^t \mathrm{d}t,\end{aligned} \tag{19}$$

where

$$\begin{aligned}\rho\left(\mathcal{N}_j, t\right) = & \exp\left[-\left(\mathcal{N}_j - \hat{\mu}_j^t\right)^2 / 2\left(\hat{\sigma}_j^t\right)^2\right] \\ & - \exp\left[-\left(\sum_i k_{imin} - \hat{\mu}_j^t\right)^2 / 2\left(\hat{\sigma}_j^t\right)^2\right].\end{aligned} \tag{20}$$

Due to page limit, we do not provide the detailed derivations in here. Interested readers may prefer to our technical report [1] for details.

For any j, we take the derivative of $\inf \mathcal{F}_j\left(\mathcal{N}_j, p_{cj}\right)$ and have

$$\begin{aligned}\frac{\partial \inf \mathcal{F}_j\left(\mathcal{N}_j, p_{cj}\right)}{\partial \mathcal{N}_j} = & -\frac{p_{cj}+2p_{lj}}{2}T \\ & + \frac{p_{cj}}{\sqrt{2\pi}}\int_0^T \frac{\left(\hat{\mu}_j^t - \mathcal{N}_j\right)}{\hat{\sigma}_j^t} \exp\left[-\frac{\left(\mathcal{N}_j - \hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}\right] \mathrm{d}t.\end{aligned} \tag{21}$$

Let the derivative equal zero and we can obtain the result $\mathcal{N}_j^*$. In conclusion, the best arrangement for OSP is $\mathcal{N}^* = \left\{\mathcal{N}_1^*, \ldots, \mathcal{N}_j^*, \ldots, \mathcal{N}_m^*\right\}$, where

$$\mathcal{N}_j^* = \arg\max_{\mathcal{N}_j} \ \inf \mathcal{F}_j\left(\mathcal{N}_j, p_{cj}\right). \tag{22}$$

Up till now, we can see that for any given p_{cj}, $\mathcal{N}_j^*$ is determined by the above function. Since we remain general forms of the distribution on $\hat{\mu}_j^t$ and $\hat{\sigma}_j^t$, it is hard for us to obtain any closed-form solution. In what follows, we design an algorithm to approach the optimality.

B. An Approximation Algorithm to OSP

The previous subsection presents a theoretical expression of $\mathcal{N}^*$. Runge-Kutta scheme [10] is an efficient method to search the optimum, and is widely applied in practice. Therefore, we propose an efficient and accurate algorithm based on the Runge-Kutta scheme.

Let $F_j\left(\mathcal{N}_j\right) = \frac{\partial \inf \mathcal{F}_j\left(\mathcal{N}_j\right)}{\partial \mathcal{N}_j}$ and the previous problem turns to finding zeros of $F_j\left(\mathcal{N}_j\right)$. A fourth order Runge-Kutta ($RK4$) scheme of $F_j\left(\mathcal{N}_j\right)$ is depicted as Algorithm 3.

In Algorithm 3, h is the step size and ε is pre-fixed small real value to control the termination of this algorithm. In every loop of this algorithm, we calculate the intermediate variables a_1, a_2, a_3 and a_4, and estimate a slope to calculate the next iteration value. According to [10], the $RK4$ based algorithm has an error bound of h^5, and the total error bound is as small as h^4. When we apply a small value of h, we can approach the result very close to the optimality.

Algorithm 3: $\hat{\mathcal{N}}_j^* = RK4\left(F_j, p_{cj}\right)$ Scheme

Input: initial-value $F_j\left(0\right), F_j' = f\left(\mathcal{N}_j, F_j\right), h, \varepsilon$
Output: $\hat{\mathcal{N}}_j^*$

```
Initialization: L_1 = L_2 = F_j(0), N_0 = 0
while |L_2| > ε do
    N_0 := N_0 + h
    Calculate F_j(N_0)
    a_1 = f(N_0, F_j(N_0))
    a_2 = f(N_0 + h/2, F_j(N_0) + (h/2) a_1)
    a_3 = f(N_0 + h/2, F_j(N_0) + (h/2) a_2)
    a_4 = f(N_0 + h, F_j(N_0) + h a_3)
    L_2 = L_1 + (h/6)(a_1 + 2a_2 + 2a_3 + a_4)
end
N̂_j^* = N_0
Return N̂_j^*
```

V. CSP's Decision on Pricing Strategy

In the previous section, the OSP decides how it implements its private cloud for any given public cloud resource pricing strategy $\mathcal{P}_c$. In this section, by knowing the OSP's best response, we answer what is the optimal pricing strategy of the CSP, such that its profit can be maximized.

A. Problem Formulation

Given the $\hat{\mathcal{N}}_j^*$ in the previous section, the CSP needs to solve $\mathcal{P}_c^* = \arg\max_{\mathcal{P}_c} \Pi_C\left(\mathcal{P}_c, \mathcal{N}^*\left(\mathcal{P}_c\right)\right)$. Therefore, we have

$$\begin{aligned}\mathcal{P}_c^* &= \arg\max_{\mathcal{P}_c} \ \int_0^T \mathcal{S}_C^t \mathcal{P}_c \mathrm{d}t \\ &= \arg\max_{\mathcal{P}_c} \ \int_0^T \left(\mathcal{S}_I^t - \mathcal{S}_L^t\right)\mathcal{P}_c \mathrm{d}t \\ &= \arg\max_{\mathcal{P}_c} \ \sum_{j=1}^m \int_0^T \left[\mathcal{S}_j^t - APP_j\left(\mathcal{N}_j, \mathcal{S}_j^t\right)\right] p_{cj} \mathrm{d}t \\ &= \arg\max_{\mathcal{P}_c} \ \sum_{j=1}^m \mathcal{H}_j\left(p_{cj}\right).\end{aligned} \tag{23}$$

The previous section gives $\mathcal{N}_j = \hat{\mathcal{N}}_j^*\left(p_{cj}\right)$. Combining Equation (17) and (18), we have

$$\begin{aligned}\mathcal{H}_j\left(p_{cj}\right) = & \int_0^T \int_{\sum_i k_{imin}}^{\hat{\mathcal{N}}_j^*\left(p_{cj}\right)} \left(s_j^t - s_j^t\right)\mathcal{G}_j\left(s_j^t\right) p_{cj} \mathrm{d}s_j^t \mathrm{d}t \\ & + \int_0^T \int_{\hat{\mathcal{N}}_j^*\left(p_{cj}\right)}^{\sum_i k_{imax}} \left[s_j^t - \hat{\mathcal{N}}_j^*\left(p_{cj}\right)\right]\mathcal{G}_j\left(s_j^t\right) p_{cj} \mathrm{d}s_j^t \mathrm{d}t.\end{aligned} \tag{24}$$

Note that the first term equals 0, because the case $APP_j\left(\mathcal{N}_j, \mathcal{S}_j^t\right) = \mathcal{S}_j^t$ implies the OSP does not need to purchase public resources. Then we can obtain the lower bound of $\mathcal{H}_j\left(p_{cj}\right)$ as

$$\begin{aligned}\inf \mathcal{H}_j\left(p_{cj}\right) = & \frac{p_{cj}}{\sqrt{2\pi}} \int_0^T \hat{\sigma}_j^t \omega\left(p_{cj}, t\right) \mathrm{d}t \\ & + \frac{p_{cj}}{2}\int_0^T \hat{\mu}_j^t \mathrm{d}t - \frac{1}{2}\hat{\mathcal{N}}_j^*\left(p_{cj}\right) p_{cj} T,\end{aligned} \tag{25}$$

Algorithm 4: $\hat{p}_{cj}^{*} = RK4S\left(\inf \mathcal{H}_j\left(p_{cj}\right)\right)$ Scheme

Input: initial-value $\inf \mathcal{H}_j\left(0\right), \left[\inf \mathcal{H}_j\left(p_{cj}\right)\right]' = g\left(p_{cj}, \inf \mathcal{H}_j\left(p_{cj}\right)\right), h, \varepsilon$

Output: $\hat{p}_{cj}^{*}$

Initialization: $L_1 = 0, L_2 = \inf \mathcal{H}_j\left(0\right), p_0 = 0$

while $|L_2 - L_1| > \varepsilon$ **do**

$p_0 := p_0 + h$

Calculate $\inf \mathcal{H}_j\left(p_0\right)$

$b_1 = g\left(p_0, \inf \mathcal{H}_j\left(p_0\right)\right)$

$b_2 = g\left(p_0 + \frac{h}{2}, \inf \mathcal{H}_j\left(p_0\right) + \frac{h}{2} b_1\right)$

$b_3 = g\left(p_0 + \frac{h}{2}, \inf \mathcal{H}_j\left(p_0\right) + \frac{h}{2} b_2\right)$

$b_4 = g\left(p_0 + h, \inf \mathcal{H}_j\left(p_{cj}\right) + h b_3\right)$

$L_1 = L_2$

$L_2 = L_1 + \frac{h}{6}\left(b_1 + 2b_2 + 2b_3 + b_4\right)$

end

$\hat{p}_{cj}^{*} = p_0$

Return $\hat{p}_{cj}^{*}$

where

$$\omega\left(p_{cj}, t\right) = \exp\left[-\left(\sum_i k_{imax} - \hat{\mu}_j^t\right)^2 / 2\left(\hat{\sigma}_j^t\right)^2\right] - \exp\left[-\left(\hat{\mathcal{N}}_j^{*}\left(p_{cj}\right) - \hat{\mu}_j^t\right)^2 / 2\left(\hat{\sigma}_j^t\right)^2\right]. \quad (26)$$

Due to the page limit, detailed derivatives are in our technical report [1]. The result we obtain is: the solution to the CSP's utility maximization problem is $\mathcal{P}_c^{*} = \left(p_{c1}^{*}, \ldots, p_{cj}^{*}, \ldots, p_{cm}^{*}\right)$, where

$$p_{cj}^{*} = \arg\max_{p_{cj}} \ \inf \mathcal{H}_j\left(p_{cj}\right). \quad (27)$$

B. Public Pricing Strategy

To further observe the feature of the solution to CSP, let us first take the derivative of $\inf \mathcal{H}_j\left(p_{cj}\right)$, so we have

$$\frac{\mathrm{d} \inf \mathcal{H}_j\left(p_{cj}\right)}{\mathrm{d}p_{cj}} = \frac{1}{\sqrt{2\pi}} \int_0^T \hat{\sigma}_j^t \omega\left(p_{cj}, t\right) \mathrm{d}t + \frac{p_{cj}}{\sqrt{2\pi}} \int_0^T \hat{\sigma}_j^t \omega\left(p_{cj}, t\right)' \mathrm{d}t + \frac{1}{2} \int_0^T \hat{\mu}_j^t \mathrm{d}t - \frac{1}{2} \hat{\mathcal{N}}_j^{*}\left(p_{cj}\right) T - \frac{1}{2}\left[\hat{\mathcal{N}}_j^{*}\left(p_{cj}\right)\right]' p_{cj} T. \quad (28)$$

A direct idea for public pricing strategy is to compute the price value such that the above equation equals zero. However, it is very hard to obtain a closed-form solution due to the complicated forms of the formulas. Therefore, we apply an efficient and accurate algorithm to approach the optimum. This algorithm also follows the $RK4$ methodology and is similar to Algorithm 3.

Let us present our approach in Algorithm 4, each iteration of which follows the $RK4$ method. Different from Algorithm 3, this algorithm searches the optimum point of $\inf \mathcal{H}_j(p_{cj})$. Similarly this algorithm has an error bound of h^5 in each step, so the overall error bound is h^4. By carefully choosing the value of h, we can guarantee a very small error bound.

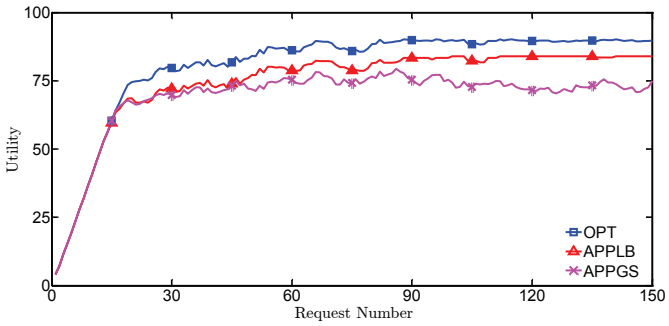


Fig. 4. Comparison of Static Algorithms

Summary. Up till now, we have proposed an integrated Stackelberg game model and presented guidelines for the OSP and the CSP to make their optimal decisions. In particular, the CSP decides the pricing scheme as $\left(p_{c1}^{*}, \ldots, p_{cj}^{*}, \ldots, p_{cm}^{*}\right)$ for each of its public cloud resources, and the OSP decides how to implement its private cloud system by deciding the amount of resources it need to prepare, i.e., $\left(\mathcal{N}_1^{*}, \ldots, \mathcal{N}_j^{*}, \ldots, \mathcal{N}_m^{*}\right)$. Due to the generality of problem formulation, as well as NP-completeness of the resource allocation problem, we do not provide closed-form solutions. Alternatively, we design efficient and accurate algorithms, and derive the theoretic formulas and bounds for these decisions.

VI. Performance Evaluation

In this section, we use real-trace data to evaluate the performance of our design. We first show the performance of our designed algorithms $APPGS$ and $APPLB$, and then compare the utility of the OSP and the CSP when using our algorithms to decide their strategies.

A. Comparison of Static Allocation Algorithms

We compare the OSP's utility when using the $APPGS$ and $APPLB$ algorithms, compared with its maximal possible utility (denoted by OPT, and obtained by exhaustive search). We assume that users' requests can be classified into three types, and each request type requires three different computing resources. We vary the request number from 1 to 150, and the proportion of each request type is fixed. Results are shown in Figure 4, where we can see that $APPGS$ achieves 85.71% utility compared to the maximal value, while $APPLB$ achieves 92.65%. We run our algorithms in a normal PC with $3.20GHz$ CPU, and the total running time of $APPGS$ and $APPLB$ are $1.05ms$ and $107.04ms$, respectively. Therefore, we can use $APPLB$ for a higher utility when computational time is allowed, or $APPGS$ for a faster calculation. In the following, we use the combination of them, denoted by APP.

B. Private Cloud Utility

In this and the next subsections, we evaluate our Algorithms 3 and 4. We consider an OSP with three different type of services. *Type 1* is a constant request over time. *Type 2* is a

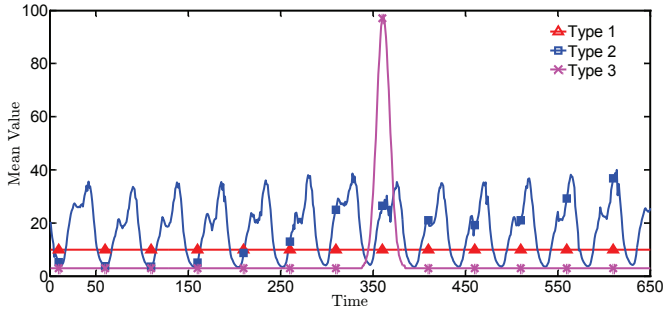


Fig. 5. Service Types

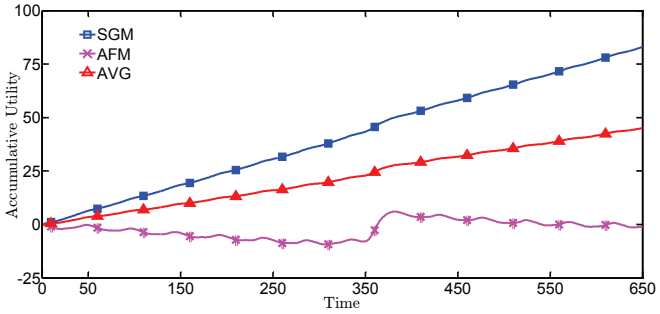


Fig. 7. Accumulative Private Utility

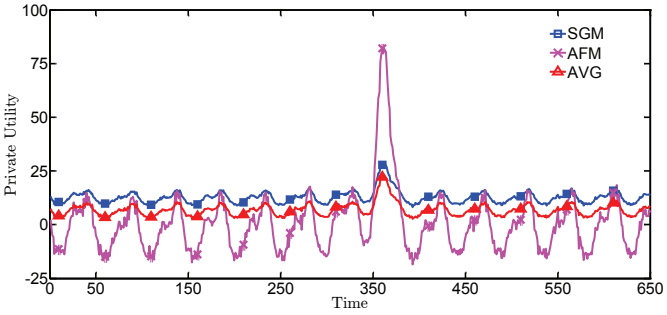


Fig. 6. Comparison of Private Utility

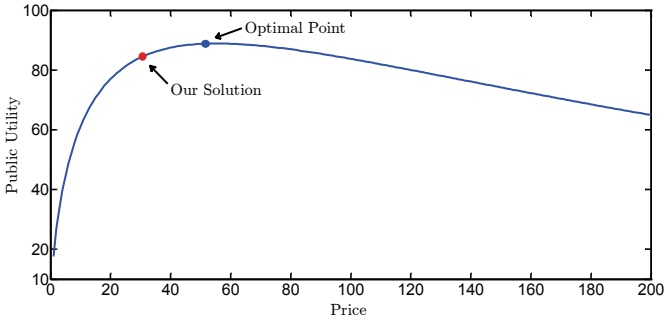


Fig. 8. Comparison of Public Utility

time-variant service with strong periodicity. *Type 3* represents a burst request (e.g., the Black Friday flooding in online sale websites). The data traces are depicted in Figure 5, where the x-axis is the time domain and y-axis represents the number of requests from users. *Type 1* and *Type 3* curves are artificially generated by us, while *Type 2* curve is a real-trace data we obtained from a particular online video service company, collected every 30 minutes from November 4 to 18, 2012. Due to the requests from that company, we anonymize the company's title, and have normalized the values.

There are numbers of related works dealing with tasks scheduling in hybrid cloud (see Section VII), but they have very different settings, and their algorithms are not comparable to ours due to different objectives. In here, we compare our design with two baseline approaches. The algorithms we consider are:

- **Stackelberg Game Model** (SGM): our approach.
- **Always-Fit-Most scheme** (AFM): the private cloud is powerful enough to deal with all requests, even at the peak time point.
- **AVeraGe scheme** (AVG): the private cloud deals with half of total requests; the other half are resorted to the public cloud.

Figure 6 shows the OSP's utility in different schemes, and the accumulative results are depicted in Figure 7. Although possessing a peak utility value, the AFM scheme has the worst performance. The reason is that the OSP has to preserve large number of idle resources, and it is a huge waste for most of the time. The AVG scheme does not consider the cost difference using private and public cloud, and thus leads to a lower utility due to high fee charged by the CSP. Our SGM scheme achieves the greatest accumulative utility, and is obviously the best of the three approaches.

C. Public Cloud Utility

In this subsection, we compare the pricing strategy of the CSP using our approach with the optimal pricing scheme. Since we need to use exhaustive search to find the optimality, we apply a simplified setting: we consider only *Type 2* requests but omit the rest. We use Algorithm 4 to reach our solution, and exhaustive search to find the optimal price. We present our results in Figure 8. We find that although our solution on the unit price is lower than the optimal choice, however, the CSP's utility under our solution reaches 94.7% to the maximal possible value. To summarize, via extensive simulations we validate that our algorithms can achieve near-optimal solutions with low time complexity.

VII. Related Work

The resource/request allocation problems in the cloud have attracted a lot of attentions. Zhen et al. [11] presented a virtualization system to dynamically allocate data center resources and support green computing. Alicherry and Lakshman [12] developed efficient resource allocation algorithms in distributed clouds. The objective is minimizing the latency between selected data centers. With a ranking mechanism, Ergu et al. [13] modeled the task-oriented problem of resource allocation in a cloud computing environment. Dán et al. considered the dynamic content allocation problem for a content delivery

system in [14], which combines cloud-based storage with low cost dedicated servers. In [15], Hao et al. proposed a non-prior method for VMs allocation in a distributed cloud to decrease the network cost. Shi et al. in [16] represented the first online combinatorial and truthful auction mechanism to model dynamic provisioning of VMs in a cloud paradigm.

Meanwhile, a number of research has been focusing on the hybrid cloud system as well. In [17], Armbrust et al. showed the hybrid cloud can achieve better performance than single public or private cloud. Bittencourt et al. showed the main characteristics of scheduling in [18], and proposed a cost optimization algorithm for the hybrid cloud in [19]. Concerning resource allocation and performance, Lee et al. proposed a cluster-based architecture to allocate resources in [20]. Authors in [21] and [22] discussed hybrid cloud management with deadline constraints. Zhou et al. [23] addressed the privacy issue in a hybrid cloud, where sensitive data are kept in trusted private cloud while insensitive data are moved to the public cloud. Beloglazov et al. [24] proposed several resource allocation policies and algorithms to realize Green IT.

Up till now, we have not found any work that combine the design of resource allocation and hybrid cloud. Different from all these previous works, we consider an online service application with specific request arrival patterns, and design the combinatorial resource allocation and pricing strategies in a hybrid cloud system.

VIII. Conclusion

In this paper, we propose a hybrid cloud design to address the heterogeneous and time varying requests in online service applications. In particular, we use a Stackelberg game model to capture the interactions between the public cloud and the private cloud, based on which we answer 1) for the online service provider, how to implement the private cloud system in terms of local resource preparation, and 2) for the public cloud provider, how to decide the optimal prices for the cloud resources so as to maximize its profit. We prove the NP-completeness of the resource allocation problem, and propose efficient and effective algorithms to approach the optimality. We show, both theoretically and empirically, that our algorithms is time-efficient and achieves near-optimal solutions. We believe this gives important guidelines to design practical hybrid cloud systems.

References

[1] "On resource allocation and pricing in hybrid cloud: A hierarchical approach," Tech. Rep., https://www.dropbox.com/s/tjwexocg5k2iktd/TechnicalReport.pdf.

[2] J. L. de Castro Silva, N. Y. Soma, and N. Maculan, "A greedy search for the three-dimensional bin packing problem: the packing static stability case," *International Transactions in Operational Research*, vol. 10, no. 2, pp. 141–153, March 2003.

[3] R. Panigrahy, K. Talwar, L. Uyeda, and U. Wieder, "Heuristics for vector bin packing," Microsoft, Tech. Rep., 2011.

[4] M. J. Osborne, *An Introduction to Game Theory*. Oxford University Press, 2004.

[5] M. Jünger, T. M. Liebling, and Denis Naddef et al., *50 Years of Integer Programming 1958-2008*. Springer, 2009.

[6] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, ser. The IBM Research Symposia Series. Springer, 1972, pp. 85–103.

[7] A. Fréville, "The multidimensional 0-1 knapsack problem: An overview," *European Journal of Operational Research*, vol. 155, no. 1, pp. 1–21, May 2004.

[8] J. Puchinger, G. R. Raidl, and U. Pferschy, "The core concept for the multidimensional knapsack problem," in *Evolutionary Computation in Combinatorial Optimization*, ser. Lecture Notes in Computer Science, vol. 3906. Springer, 2006, pp. 195–208.

[9] A. Singh, M. Korupolu, and D. Mohapatra, "Server-storage virtualization: integration and load balancing in data centers," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*.

[10] A. Jameson, W. Schmidt, and E. Turkel, "Numerical solution of the euler equations by finite volume methods using runge kutta time stepping schemes," in *Fluid and Plasma Dynamics Conference*, 1981.

[11] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107 – 1117, June 2013.

[12] M. Alicherry and T. Lakshman, "Network aware resource allocation in distributed clouds," in *IEEE INFOCOM 2012*.

[13] D. Ergu, G. Kou, Y. Peng, Y. Shi, and Y. Shi, "The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment," *The Journal of Supercomputing*, vol. 64, no. 3, pp. 835–848, June 2013.

[14] G. Dán and N. Carlsson, "Dynamic content allocation for cloud-assisted service of periodic workloads," in *IEEE INFOCOM 2014*.

[15] F. Hao, M. Kodialam, T. V. Lakshman, and S. Mukherjee, "Online allocation of virtual machines in a distributed cloud," in *IEEE INFOCOM 2014*.

[16] W. Shi, L. Zhang, C. Wu, Z. Li, and F. C. Lau, "An online auction framework for dynamic resource provisioning in cloud computing," in *ACM SIGMETRICS 2014*.

[17] M. Armbrust, A. Fox, R. Griffith, and et al., "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, April 2010.

[18] L. F. Bittencourt, E. R. M. Madeira, and N. L. S. da Fonseca, "Scheduling in hybrid clouds," *IEEE Communications Magazine*, vol. 50, no. 9, pp. 42–47, September 2012.

[19] L. F. Bittencourt and E. R. M. Madeira, "HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds," *Journal of Internet Services and Applications*, vol. 2, no. 3, pp. 207–227, December 2011.

[20] G. Lee, B. Chun, and R. H. Katz, "Heterogeneity-aware resource allocation and scheduling in the cloud," in *Proceedings of HotCloud*, 2011.

[21] R. V. den Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-optimal scheduling in hybrid IaaS clouds for deadline constrained workloads," in *International Conference on Cloud Computing (CLOUD)*, 2010.

[22] T. A. L. Genez, L. F. Bittencourt, and E. R. M. Madeira, "Workflow scheduling for SaaS/PaaS cloud providers considering two SLA levels," in *Network Operations and Management Symposium*, 2012.

[23] Z. Zhou, H. Zhang, X. Du, P. Li, and X. Yu, "Prometheus: Privacy-aware data retrieval on hybrid cloud," in *IEEE INFOCOM 2013*.

[24] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, May 2012.