Online Cost-Aware Service Requests Scheduling in Hybrid Clouds for Cloud Bursting

Yanhua Cao(✉), Li Lu, Jiadi Yu, Shiyou Qian, Yanmin Zhu, Minglu Li, Jian Cao, Zhong Wang, Juan Li, and Guangtao Xue

School of Electronic, Information and Electrical Engineering, Shanghai Jiaotong University, Shanghai, China
{buttons,luli_jtu,jiadiyu,qshiyou,yzhu,mlli,cao-jian, zhongwang,miletu,Xue-gt}@sjtu.edu.cn

Abstract. The hybrid cloud computing model has been attracting considerable attention in the past years. Due to security and controllability of private cloud, some special requests ask to be scheduled on private cloud, when requests are "bursting", the requests may be rejected because of the limited resources of private cloud. In this paper, we propose the online cost-aware service requests scheduling strategy in hybrid clouds (**OCS**) which could make suitable requests placement decisions real-time and minimize the cost of renting public cloud resources with a low rate of rejected requests. All service requests are divided into two categories, the special requests ask to be accepted on private cloud, and the normal requests are insensitive on private or public cloud. In addition, all requests arrive in random, without any prior knowledge of future arrivals. We transform the online model into a one-shot optimization problem by taking advantage of Lyapunov optimization techniques, then employ the optimal decay algorithm to solve the one-shot problem. The simulation results demonstrate that **OCS** is trade-off between cost and rejection rate, meanwhile it can let the resource utilization arbitrarily close to the optimum.

Keywords: Hybrid cloud · Service request scheduling · Cost-aware · Lyapunov optimization

1 Introduction

In recent years, the hybrid cloud model-private cloud plus public cloud [1] is getting more and more attention. It is reported by Gartner Inc, that it becomes an imperative trend of cloud computing, and more than half of large enterprises around the world will deploy hybrid clouds by the end of 2017 [2]. Generally, an enterprise has a private cloud and could elastically lease public clouds' resources when the private cloud resources are not enough. The service requests that users ask for Virtual Machines (VMs) to be hosted on the hybrid clouds, Among service requests, some are definitely specified to be processed on the private cloud with the consideration of data security. For brevity of discussion, these service requests are termed as the special requests. The other requests can be scheduled on either private or public clouds, which are called the normal requests. When service requests burst, the private cloud is incapable of responding to all requests due to the limitation of resources. In this case, the special requests will be possibly rejected, while the normal requests can be handled by renting the resources of public cloud. Generally, the higher rejection rate of the special requests, the worse service performance of the hybrid cloud, so we propose the online cost-aware service requests scheduling strategy in hybrid clouds (**OCS**) which could make appropriate requests scheduling decisions with a low rejection rate while minimize the cost of renting public cloud resources.

About service requests scheduling, Quarati et al. [3] present a set of broker strategies for hybrid clouds aimed at the execution of various instances of the weather prediction WRF model subject to different user requirements and computational conditions. It focuses on scheduling four types requests based on the reservation of a static quota of resources of private cloud and assumes that the arrivals of requests are known. This assumption is not realistic and the reservation of a static quota may result in a waste of private cloud resources. Also, Hao et al. [4] design a generalized resources placement methodology that can work across different cloud architectures, with real-time requests arrivals and departures. Maguluri and Srikant [5] propose a load balancing and scheduling algorithm in clouds that is revenue optimal, unknown the duration of service request in advance. However, they are both not for requests scheduling in hybrid clouds, so they could not consider resource limitation of private cloud and the special requests. Our goal is to minimize the cost of hybrid clouds with an acceptable rejection rate.

Fully exploring the concept of "owning the base and renting the insufficient" [6] in hybrid cloud model remains challenging in practice. First, due to security and controllability of private cloud, some special requests ask to be scheduled on private cloud, when requests are "bursting", the requests may be rejected because of the limited resources of private cloud, it is a critical challenge that guarantees low rejection rate and minimizes the cost of renting public cloud resources at the same time. Second, requests arrivals and departures, the number of requests arrivals and type of virtual machine (VM) are all random, it is difficult to predict the requests arrival rate at next time-slot. Third, the original problem model is a time average problem during the whole time-slots, it is also critical challenge to transform the original problem into a one-shot optimization problem.

In this paper, we present the **OCS**. First, our objective is to minimizes the average cost of renting public cloud resources with an acceptable rejection rate. Then we take advantage of Lyapunov optimization techniques to transform the online model into a one-shot optimization problem. Finally, we employ an optimal decay algorithm for solving the one-shot optimization problem. The simulation results demonstrate that our scheduling strategy can let the resource utilization of the private cloud arbitrarily close to the optimum on hybrid cloud scenario and minimize the cost of renting public cloud resources as well as guarantee an acceptable rejection rate when meeting "bursting".

Our main contributions are:

- We formulate the service requests scheduling problem in hybrid clouds as an optimization problem, and consider the rejection rate as the constraint condition.
- We use the Lyapunov optimization techniques to transform the original optimization problem into a one-shot optimization problem and employ an optimal decay algorithm to solve the one-shot optimization problem.
- We conduct simulations and demonstrate our scheduling strategy, and the results show that can make the resource utilization of the private cloud arbitrarily close to the optimum and be trade-off between cost and rejection rate.

The remainder of this paper is organized as follows. We present the system definition and problem model (Sect. 2). We formulate the online requests scheduling problem (Sect. 3). We discuss the evaluation results (Sect. 4). Finally, we describe the related works and conclude the paper (Sects. 5 and 6).

2 System Model

In this section, we define the formulation of the service requests scheduling problem in hybrid clouds. And then, we build an online service requests model for the problem.

2.1 Problem Definition

We now formally define the problem of the online cost-aware service requests scheduling in hybrid clouds (**OCS**). All service requests in the strategy model are divided into two categories, the special requests ask to be processed on private cloud, and the normal requests are insensitive on private or public clouds. We assume that there are multiple requests arrivals in every time-slot, and number of requests, duration of each request, and number of VM in any time-slot are all random. t indexes the time-slot. Suppose that $Rv_i = (a_{i1t}, a_{i2t}, \cdots, a_{iht}; t_i)$ is a vector that denotes the special request only accepted by the private cloud, and $i \in \{1, 2, \cdots, n_t\}$, n_t is the number of special requests in the t-th time-slot. t_i is the duration of the request Rv_i. $Ra_j = (b_{j1t}, b_{j2t}, \cdots, b_{jht}; t_j)$ is a vector that denotes the normal request accepted by the private or public cloud, and $j \in \{1, 2, \cdots, m_t\}$, m_t is the number of normal requests in the t-th time-slot. t_j is the duration of the request Ra_j. We use T_{kt} to denote the available capacity of the k resource on private cloud in the t-th time-slot. Typical resources include CPU, memory, storage disk and I/O bandwidth, when $k = 1$ denotes the resource is CPU, in the same way, $k = 2$, $k = 3$ or $k = 4$ denotes the resource is memory, storage disk or I/O bandwidth respectively. We assume that there are h classes of VMs, the amount of resource k required to host a VM of class v is given by M_{vk}, and $v \in \{1, 2, \cdots, h\}$, a_{ivt} or b_{jvt} denotes the request Rv_i or Ra_j asks for the number of VM$_v$ in the *t-th* time-slot respectively. We let the c_v denote the unit expenses of VM$_v$ on public clouds in the *t-th* time-slot. Because the private cloud belongs to the inherent property in the enterprise, we assume that the expenses of requests accepted on the private cloud is zero, so we consider the expenses of renting public cloud resources as the cost for the problem. Y_{it} and Z_{jt} are zero-one decision variables. If $Y_{it} = 1$, the special request Rv_i is accepted in the *t-th* time-slot by the private cloud, and if $Y_{it} = 0$, the request Rv_i is rejected, and we assume that the threshold of rejection rate is α. When the $Z_{jt} = 1$, the request Ra_j is accepted in the *t-th* time-slot by the private cloud, and if the $Z_{jt} = 0$, the request Ra_j is accepted by public cloud. The objective of the hybrid clouds service provider is to distribute these service requests so as to minimize its total cost. Specifically, we want the service requests scheduling decisions to be made in a dynamic manner. The problem parameters and decision variables are defined in Table 1.

2.2 Problem Modeling

The problem can be formulated as the following IP model. The cost of the hybrid clouds R_t in the *t-th* time-slot.

$$R_t = \sum_{j=1}^{m_t} \sum_{v=1}^{h} (1 - Z_{jt}) b_{jvt} c_v t_j, \tag{1}$$

From the perspective of cost, as the cost metric of an enterprise, we minimize the average cost as objection function. So the optimization problem can be formulated as follows:

Minimize:

$$\lim_{T \to +\infty} \frac{1}{T} \sum_{t=0}^{T-1} R_t \tag{2}$$

subject to:

$$Y_{it}, Z_{jt} \in \{0, 1\} \tag{3}$$

$$\sum_{i=1}^{n_t} \sum_{v=1}^{h} Y_{it} a_{ivt} v_{vk} + \sum_{j=1}^{m_t} \sum_{v=1}^{h} Z_{jt} b_{jvt} v_{vk} \le T_{kt}, \tag{4}$$

$$\lim_{T \to +\infty} \frac{\sum_{t=0}^{T-1} \sum_{i=1}^{n_t} Y_{it}}{\sum_{t=0}^{T-1} n_t} \ge 1 - \alpha \tag{5}$$

The objective function (2) represents the average cost in total T time-slots, and we get the service requests scheduling decisions in each time-slot by optimizing the problem (2). Constraint (3) gives the definitions of the decision variables. Constraint (4) ensures resources capacities of accepted requests by the private cloud are not beyond its spare capacities in each time-slot, $k \in \{1, 2, 3, 4\}$. Constraint (5) restricts average rejection rate of the special requests below the threshold value α. From the formulation, we can see that the problem is an online

Table 1. Key parameters

Notation	Definition
Rv_i	The vector denotes special request
Ra_j	The vector denotes normal request
Y_{it}	$Y_{it} \in \{0, 1\}$, when $Y_{it} = 1$, the special request is accepted by the private cloud; And if $Y_{it} = 0$ the request is rejected
Z_{jt}	$Z_{jt} \in \{0, 1\}$ when $Z_{jt} = 1$, the normal request is accepted by the private cloud; And if $Z_{jt} = 0$ the request is accepted by public cloud
h	The number of the type of VM
V	Weighting factor
α	The threshold of rejection rate
a_{ivt}	The special request asks for the number of VM_v in the *t-th* time-slot. $a_{jvt} \in [0, N]$
b_{jvt}	The normal request asks for the number of VM_v in the *t-th* time-slot. $b_{jvt} \in [0, N]$
t_i	Duration for service request
c_v	The unit expense of VM_v in public cloud
M_{vk}	The k resource capacity of VM_v
T_{kt}	The k resource total spare volume of private cloud in the *t-th* time-slot, $k \in \{1, 2, 3, 4\}$
n_t	The number of the special requests in *t-th* time-slot
m_t	The number of the normal requests in *t-th* time-slot
Ns	The noted solution
d	The queue of decay sequences

service requests scheduling, and it is very difficult to solve the problem (2). We transform the original online model into a one-shot optimization problem by taking advantage of Lyapunov optimization techniques.

3 Online Cost-Aware Service Requests Scheduling Strategy

We consider that there are multiple requests in each time-slot, and the requests arrivals are random. The objective function is to minimize the average cost of renting the public cloud resources. The constraint (4) guarantees that the resource amount of accepted requests below the total capacity of spare k resource in the *t-th* time-slot, and the constraint (5) however implies that the average rejection rate of the special requests must not exceed the given threshold value α. During the $t + 1$-*th* time-slot, the below Eq. (6) records the number of the rejected special requests. As a result, a closer examination on the constraint suggests that the inequality (5) can be modeled as a queue $H(t)$. When the rejected number $H(t)$, that is the length of the queue, is stable. We can take advantage of Lyapunov optimization techniques [7] to transform the problem (2) into a one-shot optimization problem.

3.1 Problem Transformation by Lyapunov Optimization

Based on the above analysis, in order to capture the number of rejected special requests which is above the threshold value, we introduce the virtual queue $H(t)$, which is used to accumulate the part of the rejected number that exceeds the maximum permitted number in the *t-th* time-slot. Initially, we define $H(0) = 0$ and then update the queue according to Eq. (6), inequality (5) guarantees the virtual queue $H(t)$ is bounded. Whereas, if the virtual queue $H(t)$ is unbounded, more and more special requests are rejected, worse and worse service performance of the hybrid clouds is, under the circumstance, it is necessary to upgrade the infrastructure of private cloud for improving performance.

$$H(t+1) = max\{H(t) + n_t(1-\alpha) - \sum_{i=1}^{n_t} Y_{it}, 0\} \tag{6}$$

Lemma 1. The queue $H(t)$ is stable. $\lim_{T\to+\infty} \frac{H(T)}{T} = 0$.

In order to save space, the proof process is omitted here. Taking advantage of Lyapunov optimization techniques, we consider the Lyapunov function $L(H(t))$ as follows:

$$L(H(t)) = \frac{1}{2}H^2(t). \tag{7}$$

The larger of the $H(t)$, the more number of rejected special requests, the constraint (5) is hard to meet, specifically, the average rejection rate exceeds the threshold α. Therefore, it is essential to keep the average rejection rate being below the threshold value. First we define the one-slot Lyapunov drift of $L(H(t))$ as follows:

$$\Delta(L(H(t))) = E\{L(H(t+1)) - L(H(t))|H(t)\}. \tag{8}$$

Lemma 2. For any $t \in [0, T-1]$, given any possible control decision, the Lyapunov drift $\Delta(L(H(t)))$ can be deterministically bounded as follows:

$$\Delta(L(H(t))) \le \frac{1}{2}n_t^2\alpha^2 + H(t)E[n_t(1-\alpha) - \sum_{i=1}^{n_t} Y_{it}|H(t)]. \tag{9}$$

In order to save space, the proof process is omitted here. According to the Lemma 2, we get the upper bound of $\Delta(L(H(t)))$, after bounding the one-slot Lyapunov drift, we can minimize the upper bound to meet the constraint (5). If we add a penalty term VR_t to both sides of the inequality (9).

$$VR_t + \Delta(L(H(t))) \leq \frac{1}{2}n_t^2\alpha^2 + VR_t + H(t)E[n_t(1-\alpha) - \sum_{i=1}^{n_t} Y_{it}|H(t)]. \quad (10)$$

So, the underlying objective of our optimal scheduling decisions is to minimize the bound on the following expression in one-shot optimization problem, that is the each time-slot problem. And the problem (2) can be transformed as follows: Minimize:

$$VR_t + H(t)[n_t(1-\alpha) - \sum_{i=1}^{n_t} Y_{it}] \quad (11)$$

s.t. constraints (3) and (4).

Until now, we have transformed the long-term optimization problem to the above optimization problem in each time-slot. During each time-slot $t \in \{0, 1, \cdots, T-1\}$, we minimize the amount of rejected special requests and the cost of renting public cloud resources. And V is a weighting factor to adjust our emphasis on the rejected amount and the cost of renting public cloud resources, by tuning it, we can choose which objective we need to underline.

3.2 The Analysis of the Relationship Between Cost of Renting and Rejected Requests Number

We now analyze the relationship between cost of renting and number of rejected requests, with any parameter V such that $V > 0$, we assume that R_{op} is the optimal cost of renting in theory during any time-slot $t \in \{0, 1, \cdots, T-1\}$.

$$\frac{1}{T}\sum_{t=0}^{T-1} R_t = R_{op}. \quad (12)$$

Based on the constraint (5), we have

$$\frac{\sum_{t=0}^{T-1}\sum_{i=1}^{n_t}(1-Y_{it})}{\sum_{t=0}^{T-1} n_t} \leq \alpha. \quad (13)$$

Due to α is an upper bound of inequality (13), there must exist a β, and $\beta > 0$, which can make the inequality (13) to the following form

$$\frac{\sum_{t=0}^{T-1}\sum_{i=1}^{n_t}(1-Y_{it})}{\sum_{t=0}^{T-1} n_t} \leq 1 - \alpha - \beta. \quad (14)$$

The problem (11) implies that we should make decisions to minimize right side of inequality (10). By applying R_{op} to inequality (10), and summing up the inequality (10) over time-slot $t \in \{0, 1, \cdots, T-1\}$, and then dividing both sides by T, we have

$$\begin{aligned} &\frac{V}{T}\sum_{t=0}^{T-1} R_t + \frac{L(H(T)) - L(H(0))}{T} \\ &\le \frac{1}{2}n_t^2\alpha^2 + VR_{op} - \frac{\beta}{T}\sum_{t=0}^{T-1} H(t). \end{aligned} \tag{15}$$

Due to the $H(t)$ stability, we get $\lim_{T\to\infty} \frac{H(t)}{T} = 0$,
$H(t)$ is bounded, so $\lim_{T\to\infty} \frac{L(H(t))-L(H(0))}{T} = 0$.
Due to the $-\frac{\beta}{T}\sum_{t=0}^{T-1} H(t) < 0$, we have

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} R_t \le \frac{n_t^2\alpha^2}{2V} + R_{op}. \tag{16}$$

Meanwhile, note that $\frac{V}{T}\sum_{t=0}^{T-1} R_t > 0$, we have

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} H_t \le \frac{\frac{1}{2}n_t^2\alpha^2 + VR_{op}}{\beta}. \tag{17}$$

The inequalities (16) and (17) prove that the relationship cost of renting and number of rejected requests is tradeoff. The inequality (16) shows that the average cost of renting under the online service requests scheduling can be pushed closer to the optimum value R_{op} when the V is very large. However, if the V is too large, the $H(t)$ will be unstable, which means that the rejection rate of special requests exceeds the threshold value α. And if the V is too small, the cost of renting public clouds become very large. So within the above online service requests scheduling strategy, tuning the parameter V such that $V > 0$ for all $t \in \{0, 1, \cdots, T-1\}$, we can minimize both the cost of renting and the number of rejected requests.

3.3 Optimal Decay Algorithm for the Zero-One Integer Linear Programming

Toward the problem (11), it is a representative zero-one integer linear programming problem. Generally, solving the problem by the enumeration method or branch and bound method [8], there are some special methods for the problem, such as for the assignment problem with the Hungarian method. However, there are unknown number of requests in each time-slot for our problem, the performance of solving the problem lies on the amount of requests in each time-slot. When the amount is very huge, the performance of these traditional methods of solving the problem is not good and the algorithm complexity is $O(2^n)$ (n is the number of decision variables). So it is crucial to improve the speed of solving the problem (11). Zhao et al. [9] proposes cards-flipping algorithm for Zero-one integer linear programming, which can examine the former S_i possible solutions in turn and without traversing the objection function values of all possible solutions, and if the S_i-*th* is the first feasible solution, the solution is optimal, without any calculation for the rest of $2^n - S_i$ possible solutions, the total amount of calculations is S_i. Based on [9], we propose the optimal decay algorithm (**ODA**) for the problem (11). The processes of solving the problem as follows: Firstly, getting the optimal solution of objective function without considering any constraint conditions, this is the loosest solution, we can call it as the noted solution Ns. And secondly, validating the noted solution whether meets the constraints (3) and (4). If not, we get a new noted solution Ns in terms of decay sequence d_0 with minimum decay value in the decay queue d instead of the pre-noted solution. The last, validating it whether satisfies the constraints (3) and (4) or not, if so, the new Ns is the optimal solution, if not, continuing to validate the next new noted solution according to the new d_0 until we find the optimal solution. Refer to Algorithm 1 for details.

d is the queue of decay sequences, and decay sequence is a sequence of 0 and 1, the initial d is $\{(0, \cdots, 0, 1)\}$. Such as there are three decision variables x_1, x_2, x_3 and their coefficients are a_1, a_2, a_3 respective in a minimization problem, if $|a_2| \geq |a_1| \geq |a_3|$, the Cs is the indexes sequence of the sorted decision variables, $Cs = (2, 1, 3)$. If a decay sequence $d_i = (0, 1, 1)$, its decay value $dv_i = |a_1| + |a_3|$. The d_i can be transformed into $d_{i+1} = (1, 0, 1)$ and $d_{i+2} = (1, 1, 1)$ [9].

Lemma 3. The noted solution is the optimal solution.

Proof. **The objective function is minimized**
Assuming the noted solution is Ns, and the *i-th* element is Ns_i, and a_i is its coefficient, R_n is the objection function value of the problem (11).

$$Ns_i = \begin{cases} 0 & a_i > 0 \\ 1 & a_i < 0. \end{cases} \tag{18}$$

$Ns^{'}$ is the any solution (not noted solution), and the *i-th* element is $Ns^{'}_i$, R is the objection function value of any solution. We assume the $Ns_i \neq Ns^{'}_i$. (otherwise, the any solution is noted solution)

① if $a_i > 0$:
In term of the equation (18), the $Ns_i = 0$;
Due to the $Ns_i \neq Ns^{'}_i$, the $Ns^{'}_i = 1$;
So $R_n - R = a_i Ns_i - a_i Ns^{'}_i = 0 - a_i = -|a_i|$.

② if $a_i < 0$:
In term of the equation (18), the $Ns_i = 1$;
Due to the $Ns_i \neq Ns^{'}_i$, the $Ns^{'}_i = 0$;
So $R_n - R = a_i Ns_i - a_i Ns^{'}_i = a_i - 0 = -|a_i|$.
In summary, in term of ① and ②, the $R_n < R$.

For the any $Ns_i \neq Ns^{'}_i$ the objection function value of any solution is always greater than the noted solution's, so the objection function value of noted solution is the minimum value, and the noted solution is the optimal solution.

Algorithm 1. Optimal Decay Algorithm for the 0-1 Integer Linear Programming

```
Input:   $T_{kt}$, $Rv_i$, $Ra_j$, $1 \le i \le n_t$, $1 \le j \le m_t$;
Output:   $Ns = (Y_{1t}, \cdots, Y_{n_t}, Z_{1t}, \cdots, Z_{m_t})$;
 1: get the initial $d$ and the noted solution
    $Ns = [1, 1, \cdots, 1]$
 2: if $Ns$ meets Constraint (3) and (4) then
 3:    $Ns$ is the optimal solution
 4: end if
 5: if $Ns$ does not meet constraint (3) and (4) then
 6:    repeat
 7:       for $i = 0$ to $n_t + m_t$ do
 8:          if $(d_0[i] == 1)$ // $d_0$ is the minimization decay sequence in the queue $d$
 9:          $ind = Cs[i]$ // $Cs$ is the index sequence of the sorted decision variable
10:          $Ns[ind] = abs(Ns[ind] - 1)$ // $ind$ is the index of decision variable
11:       end for
12:       if $Ns$ does not meet Constraint (3) and (4) then
13:          transform $d_0$ into $d_1$ and $d_2$, and then delete $d_0$ from the queue $d$
14:          calculate decay values $dv_1$ and $dv_2$ and insert them according to their decay
             values ascending into $d$
15:       end if
16:    until $Ns$ meets Constraint (3) and (4) or $d$ is empty
17:    if $Ns$ meets Constraint (3) and (4) then
18:       $Ns$ is the optimal solution
19:    end if
20:    if $d$ is empty then
21:       the problem (14) no solution
22:    end if
23: end if
```

Lemma 4: The decay value of d_0 is minimum during no verified decay sequences every time.

Proof. Assuming any decay sequence
$d_i = (q_1, \cdots, q_{i-1}, 1, 1, 0, \cdots, q_n)$,
$q_i = 1$, $q_{i+1} = 1$, the others are 0, $w = (b_1, \cdots, b_n)$,
$b_1 \ge \cdots \ge b_n \ge 0$, $i \in \{1, \cdots, n\}$, $\odot$ is vector inner product.
the d_i is transformed into the below two decay sequences d_{i+1} and d_{i+2},
$d_{i+1} = (0, \cdots, 1, 0, 1, 0, \cdots, 0)$,
$d_{i+2} = (0, \cdots, 1, 1, 1, 0, \cdots, 0)$,
$dv_{i+1} - dv_i = w \odot d_{i+1} - w \odot d_i = b_{i+1} - b_i$,
so $dv_{i+1} \ge dv_i$. $dv_{i+2} - dv_i = w \odot d_{i+2} - w \odot d_i = b_{i-1}$,
so $dv_{i+2} \ge dv_i$.

Due to the decay value of d_0 is minimum in the queue d and the decay values of other decay sequences are not in the queue d and are greater than any decay values in d, so the decay value of d_0 is minimum during no verified decay sequences every time.

In term of Algorithm 1, the total number of decay sequences is 2^{n_t} in the *t-th* time-slot, so this algorithm can traverse all possible solutions in the worst case. Based on the Lemmas 3 and 4, we know that the algorithm do not need to traverse all possible solutions, and it only calculate and validate the former S_i possible solutions, and without any calculations for the rest of $2^{n_t} - S_i$ possible solutions, so the computational complexity is $O(S_i)$ and reliability is equivalent to enumeration method.

4 Performance Evaluation

In this section, we conduct the simulations to evaluate the performance of the online cost-aware service requests scheduling in hybrid clouds. due to without any priori knowledge of future request arrival pattern, amount of resources and types, and service time, so we use synthetic resources requests generated by statistical function.

4.1 Simulation Setup

We set the capacity of private cloud: 100 vCPUs, 300 Gb RAM, 2000 Gb disk storage, and the prices of VMs are based on Aliyun's pricing model.

Requests Arrival Pattern: We assume the resource requests arrival pattern follows the poisson arrival; The amount of resources and types, and service time are all generated randomly by statistical function. In addition, we also design two other request arrival patterns, one is that the request arrival pattern follows an uniform distribution [10] and the amount of VM and service time of every request also follow an uniform distribution, the other is that the number of requests arrivals is constant and the amount of VMs and service time of every request are also constant [3].

Two Requests Scheduling Strategies for Comparison: In order to explore the performance of the online service requests scheduling in hybrid clouds. We set two simple strategies to be compared with the online cost-aware service requests scheduling in hybrid clouds (**OCS**). One is the "public cloud first", which tends to response all the normal requests to public cloud. The other strategy is the "private cloud first" strategy, it allows to respond the all resource requests to the private cloud as many as possible, which is a totally contrary strategy to "public cloud first". By comparing **OCS** with such two extreme strategies, we can analyze the performance of **OCS** more clearly.

4.2 Performance Evaluation

Cost of Renting. We apply three different optimizing strategies to three request arrival patterns. And the results are displayed in Fig. 1. The request arrival pattern is introduced in Sect. 4.1. In Fig. 1, we can observe that the average cost of the "public cloud first" strategy is the maximum among the three strategies, because all requests except the special requests are all scheduled to the public cloud, the cost of all normal requests is maximum. While the "private cloud first" strategy can enable the cost is minimum, because of all requests scheduled to the private cloud first if the resources of private cloud are enough for these requests, and the cost of resources of the private cloud is free due to the private cloud with enterprise's own property, so it is the optimal strategy of cost minimization. However, in our problem, the "private cloud first" strategy is not good because it makes more number of rejected requests in Fig. 2. Based on Figs. 1 and 2, although it seems that **OCS** is not outstanding in either cost of renting or number of rejected requests, we can discover that **OCS** is trying to make a tradeoff between the cost of renting and number of rejected requests.

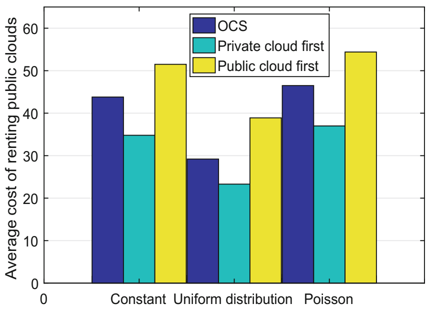

Fig. 1. Average cost of renting public cloud resources

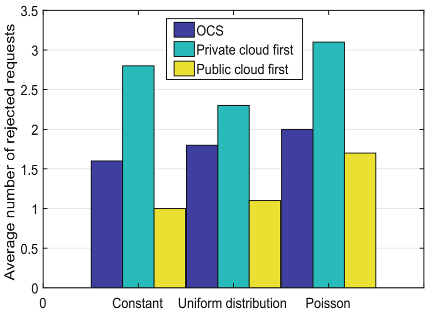

Fig. 2. Average number of rejected requests

Number of Rejected Requests. As illustrated in Fig. 2, the average number of "public cloud first" strategy is minimum, because the "public cloud first" strategy makes the private cloud only to accept the special requests but the normal requests are all scheduled to the public cloud. On the contrary, the average number of "private cloud first" strategy is maximum among the three strategies, due to all requests first scheduled to the private cloud, the resources of private cloud are quickly used up, leading to more special requests rejected. If the objective of problem is to minimize rejection rate, the "public cloud first" is optimal strategy. However, our objective is to minimize the cost of renting public cloud resources, and the average number and cost of renting of **OCS** strategy are both between the above two strategies. Based on Figs. 1 and 2, the average cost of "private cloud first" strategy is minimum but the average number of rejected requests is maximum, and the number of "public cloud first" strategy is minimum yet the average cost of renting public cloud resources is maximum. However, the cost and number of **OCS** are both between the above two strategies, it seems that to cost a little bit more for less number of rejected requests, the **OCS** is searching the balancing point of service quality between the "public cloud first" and "private cloud first".

Resource Utilization of Private Cloud. Figures 3, 4 and 5 show the resource utilization of "public cloud first" strategy is almost always lower than the other

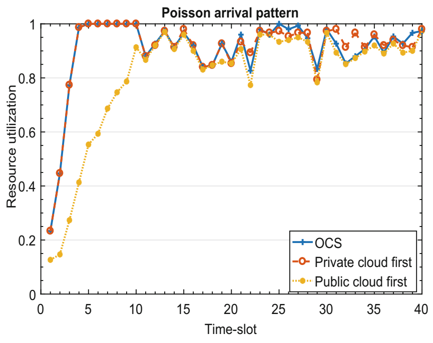

Fig. 3. Resource utilization with poisson arrival pattern

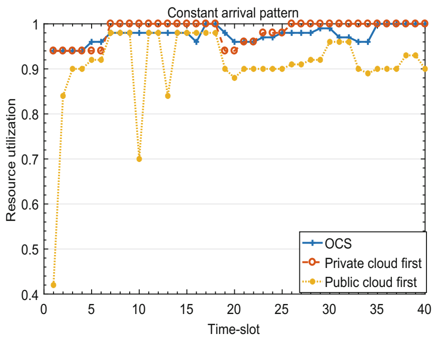

Fig. 4. Resource utilization with uniform distribution arrival pattern

two strategies over time-slots, although the strategy guarantees the lowest rejection rate, it will cause the waste of resources of private cloud. On the contrary, the resource utilization of "private cloud first" strategy is almost highest among the three strategies over time-slots, it is very clearly that any requests are scheduled to the private cloud first if the resources of private cloud are enough. However, the resource utilization of **OCS** strategy is nearly the same with "private cloud first" strategy over time-slots, the **OCS** strategy can minimize the cost of renting public cloud resources under guaranteeing the low rejection rate, meanwhile it can make full use of the private cloud resources. **The cost of renting and number of rejected requests tradeoff.** To analysis the trade-off between the cost of renting public cloud resources and number of rejected requests, we vary the parameter V to choose the metric we want to emphasize, average cost of renting and average number. In Fig. 6, by tuning the value of V to a small one, we can observe that the average number of rejected requests is small while the cost of renting is large. Meanwhile, by setting a large value of V, it brings markedly increase of the average number and decrease of the average cost of renting. Furthermore, when the average cost drops, the average number grows remarkably. And choosing a value of V represents how much we emphasize the average cost compared to the average number of rejected requests and finally get most suitable V to trade-off the average cost of renting public cloud resources and number of rejected requests.

5 Related Work

For scheduling in hybrid clouds, there are main two aspects, scheduling service requests and application tasks. And some works consider the problem of service requests scheduling in clouds [3–5]. We discuss the service requests scheduling problem in this paper, and we recognize that only the work [3] that is similar to our work. [3] takes four types of user resources requests into consideration in hybrid clouds, our study differs in at least three important aspects. First,

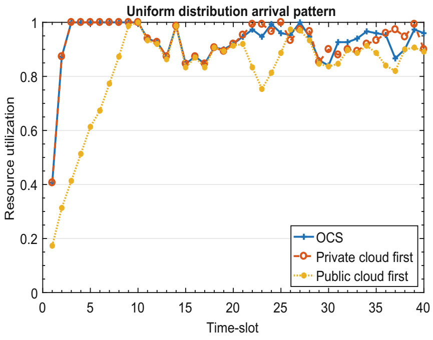

Fig. 5. Resource utilization with constant arrival pattern

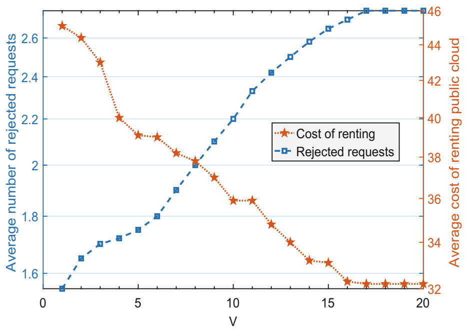

Fig. 6. The cost of renting and number of rejected requests trade-off

we model the user service requests scheduling in the hybrid clouds and employ Lyapunov optimization techniques to transform the objection function. Second the model is suitable for all kinds of users service requests. Third, our work proposes an online service requests to schedule the user requests in hybrid clouds. However, [3] only analyzes the historic data and the average number of rejected requests, and the rejection rate can not be guaranteed. Currently, there are a lot of works [11–13] about the application tasks scheduling in clouds, and [13] takes straggler tasks into consideration the tasks scheduling in cloud, our next work will study the straggler tasks in hybrid clouds.

A majority of the existing literatures leverage Lyapunov optimization techniques. For example, with respect to cloud bursting, [6] takes advantage of Lyapunov optimization techniques to design an online decision algorithm for requests scheduling. [14] proposes dynamic service migration and workload scheduling in edge-clouds, and models a sequential decision making markov decision problem, then use the technique of Lyapunov optimization over renewals. In addition, some works [15,16] apply the auction scheduling method in cloud. In this paper, we use the techniques of Lyapunov optimization to transform the average time problem to optimization problem in each time-slot.

6 Conclusion

With increasing the hybrid cloud model in enterprises, due to the limited resources of private cloud, it is necessary to consider the service requests scheduling between private cloud and public cloud. In this paper, we design an online algorithm to schedule users' service requests, meanwhile, the method minimizes the cost of renting public cloud resources with an acceptable rejection rate. By applying Lyapunov optimization techniques, we can make scheduling decisions for all requests in each time-slot. Furthermore, we employ the optimal decay algorithm for the zero-one integer linear programming to solve the optimization problem. The simulation results demonstrate that our scheduling strategy can make the resource utilization of the private cloud arbitrarily close to the optimum and be trade-off between cost of renting public cloud resources and rejection rate of the special requests.

Acknowledgments. This work has been funded by High-Tech Research and Development Program of China under grant No. 2015AA01A202 and partly supported by NSFC (Nos. 61170238, 61420106010).

References

1. Hoseinyfarahabady, M.R., Samani, H.R., Leslie, L.M., Lee, Y.C., Zomaya, A.Y.: Handling uncertainty: pareto-efficient BoT scheduling on hybrid clouds. In: 2013 42nd International Conference on Parallel Processing, pp. 419–428. IEEE (2013)
2. Sill, A.: Socioeconomics of cloud standards. IEEE Cloud Comput. **2**(3), 8–11 (2015)
3. Quarati, A., Danovaro, E., Galizia, A., Clematis, A., D'Agostino, D., Parodi, A.: Scheduling strategies for enabling meteorological simulation on hybrid clouds. J. Comput. Appl. Math. **273**, 438–451 (2015)
4. Hao, F., Kodialam, M., Lakshman, T., Mukherjee, S.: Online allocation of virtual machines in a distributed cloud. In: IEEE INFOCOM 2014-IEEE Conference on Computer Communications, pp. 10–18. IEEE (2014)
5. Maguluri, S.T., Srikant, R.: Scheduling jobs with unknown duration in clouds. In: INFOCOM, 2013 Proceedings IEEE, pp. 1887–1895 (2014)
6. Niu, Y., Luo, B., Liu, F., Liu, J., Li, B.: When hybrid cloud meets flash crowd: towards cost-effective service provisioning. In: 2015 IEEE Conference on Computer Communications (INFOCOM), pp. 1044–1052. IEEE (2015)
7. Neely, M.J.: Stochastic network optimization with application to communication and queueing systems. Synth. Lect. Commun. Netw. **3**(1), 1–211 (2010)
8. Mavrotas, G., Diakoulaki, D.: A branch and bound algorithm for mixed zero-one multiple objective linear programming. Eur. J. Oper. Res. **107**(3), 530–541 (1998)
9. Zhao, N., Wei-Jian, M.I., Wang, D.S.: A new algorithm for zero-one integer linear programming. Oper. Res. Manag. Sci. **21**(5), 111–118 (2012)
10. Zheng, L., Joe-Wong, C., Tan, C.W., Chiang, M., Wang, X.: How to bid the cloud. ACM SIGCOMM Comput. Commun. Rev. **45**(4), 71–84 (2015)
11. Dabbagh, M., Hamdaoui, B., Guizani, M., Rayes, A.: Efficient datacenter resource utilization through cloud resource overcommitment. In: 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 330–335. IEEE (2015)
12. Huang, Z., Balasubramanian, B., Wang, M., Lan, T., Chiang, M., Tsang, D.H.: Need for speed: cora scheduler for optimizing completion-times in the cloud. In: 2015 IEEE Conference on Computer Communications (INFOCOM), pp. 891–899. IEEE (2015)
13. Ren, X., Ananthanarayanan, G., Wierman, A., Yu, M.: Hopper: decentralized speculation-aware cluster scheduling at scale. In: ACM Conference on Special Interest Group on Data Communication, pp. 379–392 (2015)
14. Urgaonkar, R., Wang, S., He, T., Zafer, M., Chan, K., Leung, K.K.: Dynamic service migration and workload scheduling in edge-clouds. Perform. Eval. **91**, 205–228 (2015)

15. Zhou, Z., Liu, F., Li, Z., Jin, H.: When smart grid meets geo-distributed cloud: an auction approach to datacenter demand response. In: 2015 IEEE Conference on Computer Communications (INFOCOM), pp. 2650–2658. IEEE (2015)
16. Zhang, X., Wu, C., Li, Z., Lau, F.C.M.: A truthful (1-ε)-optimal mechanism for on-demand cloud resource provisioning. In: 2015 IEEE Conference on Computer Communications (INFOCOM), pp. 1053–1061. IEEE (2015)