Shen W. et al. Combined Cloud: A Mixture of Voluntary Cloud and Reserved Instance Marketplace. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY : 1– June 2016

Combined Cloud: A Mixture of Voluntary Cloud and Reserved Instance Marketplace

Wei Shen[1], *Student Member*, *IEEE*, Wanchun Dou[1,*], *Member*, *IEEE*, Fan Wu[2],*Member*, *IEEE*, Shaojie Tang[3], *Member*, *IEEE* and Qiang Ni[4], *Senior Member*, *IEEE*

[1]*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210046, China.*
[2]*Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai 200240, China*
[3]*Department of Information Systems, University of Texas at Dallas, TX 75080, USA.*
[4]*School of Computing and Communications, InfoLab21, Lancaster University, Lancaster LA1 4WA, UK.*

E-mail: shenwei0917@126.com; douwc@nju.edu.cn; fwu@cs.sjtu.edu.cn; tangshaojie@gmail.com; q.ni@lancaster.ac.uk

Abstract Voluntary Cloud is a new paradigm of cloud computing. It provides an alternative selection along with some well-provisioned clouds. However, for the uncertain time span that participants share their computing resources in Voluntary Cloud, there are some challenging issues, i.e., fluctuation, under-capacity and low-benefit. In this paper, an architecture is first proposed based on Bittorrent protocol. In this architecture, resources could be reserved or requested from reserved instances marketplace and could be accessed with a lower price in a short circle. Actually, these resources could replenish the inadequate resource pool and relieve the fluctuation and under-capacity issue in Voluntary Cloud. Then, the fault rate of each node is used to evaluate the uncertainty of its sharing time. By leveraging a linear prediction model, it is enabled by a distribution function which is used for evaluating the computing capacity of the system. Moreover, the cost optimization problem is investigated and a computational method is presented to solve the low-benefit issue in Voluntary Cloud. At last, the system performance is validated by two sets of simulations. And the experiment results show the effectiveness of our computational method for resource reservation optimization.

Keywords Voluntary Cloud, System Architecture, Resource Reservation

1 Introduction

"Time-sharing", as a concept in computer field, has been proposed by John McCarthy in 1990s, which is amazing the world by a new paradigm, "Cloud Computing" nowadays. "Time-sharing" is no longer a compromise with scarce computing resources but a pursuit for

a more efficient and economical computing model. With prevalence of Cloud Computing [1][2][3][4], almost all the well-known companies are devoted to establishing many datacenters all over the world and providing the public strong and stable cloud services. However, as the cloud customers are pure consumers, their local resources have been largely ignored especially considering the fast growing of personal computing capacities [5]. Thus, a novel self-organizing cloud called Voluntary Cloud has emerged as a compelling paradigm, which brings many advantages such as cheapness and diversity appealing to the public. Unlike grid computing [6], each participant can share "spare time" of their physical machine with each other in a real Voluntary Cloud named Spot Cloud[1]. Yet, the uncertainty of sharing time is a critical issue in this model. When a part of participants are doing some computing tasks, some exceptions may happen in the entire execution. For example, some physical machines are broken down or some of them need to complete an urgent task for themselves. In [7], the authors try to solve this problem by establishing a fault-tolerant architecture named BFTCloud. Although the mature fault-tolerant architecture has been proved to be successful in deploying distributed services where sufficient resource pool can replenish the fault nodes, the problem may be different in immature or growing-up cloud services. There are some challenging issues in a small scale Voluntary Cloud: 1) fluctuation: the uncertainty of sharing time results in a high fault rate of each working node, which brings large fluctuation to computing ability of the Voluntary Cloud. Usually, this fluctuation makes computing ability of the system hardly estimated. Specially, when the system accepts a series of tasks which is filled with almost all the nodes and some exceptions happen on some nodes at the same time, many of these tasks cannot be completed on time; 2) under-capacity: indeed, some fluctuations do not bring a big problem to some large cloud providers where the resource pool is big enough. But, in a small scale Voluntary Cloud, the insufficient resources cannot resist drastic fluctuations; 3) low-benefit: if a task needs n nodes to complete, the system should prepare a resource pool where the replicas are more than $3n + 1$ to ensure the tasks' smooth implementation in BFTCloud architecture [7]. It improves the system stability but loses over 2/3 system performances and economic benefits.

Reserved Instance Marketplace is a platform firstly proposed by Amazon which can help the reserved instance owners sell the remainder of their reserved resources [8]. In the Reserved Instance Marketplace, resources could be reserved with a lower price in a short circle, which is a good supplement to Voluntary Cloud. Thus, as a Voluntary Cloud organizer, in order to improve the system stability and economic benefit, he/she could reserve the computing resources from the markets like Reserved Instance Marketplace[2] or even request some resources from some well-provisioned clouds. It not only ensures the stability of Voluntary Cloud but also supplies sufficient elasticity for end-users. Just as illustrated

[1]http://www.virtustream.com/, Sep 2016

[2]http://aws.amazon.com/cn/ec2/purchasing-options/reserved-instances/marketplace/ Sep 2016

by Fig. 1, this organization combined resources from Voluntary Cloud, Reserved Instance Marketplace and some well-provided clouds, therefore, it is called "Combined Cloud". In summary, this paper makes the following contributions: 1) We identify three challenging issues in the existing system architectures for Voluntary Cloud and propose a framework based on Bittorrent protocol to solve them. 2) We propose a computational method to calculate the quantity of reserved resources to optimize our economic benefit on the basis of ensuring the correct implementation of the system. 3) We conduct two sets of simulations to test our system performance and verify the effectiveness of our computational method for resource reservation optimization.

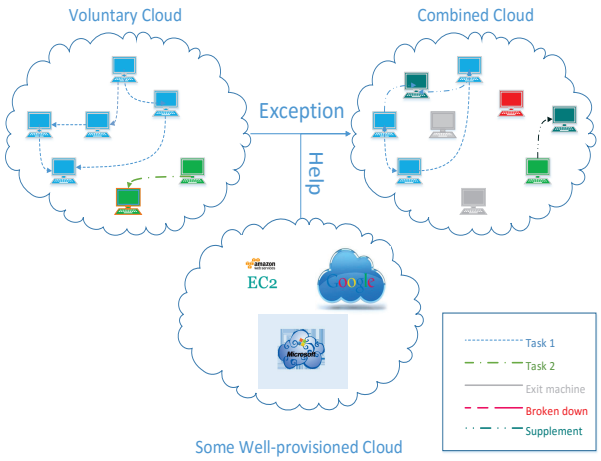


Fig. 1: Combined Cloud

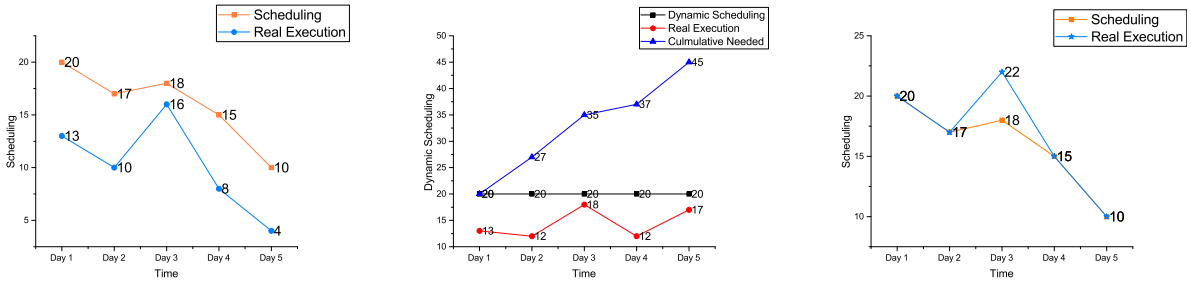
The rest of this work is organized as follows, a motivating example is proposed to demonstrate the existing problems in Voluntary Cloud and then an architecture is designed based on Bittorrent protocol to solve these problems in Section 2. In Section 3, a computational method is proposed to calculate the appropriate number of reserved resources to optimize system economic benefits. In Section 4, two sets of simulation experiments are conducted to verify the effectiveness of our computational method for resource reservation optimization. At last, some related work is discussed in Section 5 and the conclusion is made in Section 6.

2 System Architecture

In this section, a motivating example is proposed to specify the research problems in existing architecture for Voluntary Cloud. Suppose some customers' resources are organized to be a small scale Voluntary Cloud and the maximum available capacity of each customer is assumed to be 20 computing resources per day. There are two tasks waiting to be assigned, one of which needs to complete 30 parts computing tasks in 4 days and the other needs to complete 50 parts in 5 days. Assume that the expected schedule plan is shown by the "sche" column in Table 1. However, there are some exceptions in the real execution which could result in fault-tolerant problem and scalable problem.

2.1 Fault-tolerant Problem in Voluntary Cloud

In the real execution, some executing tasks would be aborted by the volunteer due to some unexpected emergencies. Therefore, this uncertainty of sharing time results in that the tasks would not be all completed as scheduled. Specifically in Day 1, the nodes which implement Task 1 only complete 6 parts computing task. Thus it is easy to find that the task would



(a) Task scheduling and execution (b) Task dynamic scheduling (c) Task scalable problem

Fig. 2: The comparison of scheduling and execution in the motivating example

not be completed as scheduled just as shown in Fig. 2(a).

In Fig. 2(b), even if the system adopts dynamic scheduling method and allocates the spare resources to replenish preceding fault nodes, the red line which indicates the cumulative needed resources (the sum of the scheduled resources and the supplement for preceding fault nodes) surpasses the green line which shows the real execution. Thus, although the so-called dynamic scheduling method that represented by blue line adopts all available resources, the result is not satisfactory. Undoubtedly, there are many other better methods to solve this problem, for example the system can only accept the task 1 and refuse task 2. However, no matter what the methods adopted, the system would waste some computing resources (In this example, if you only accept the task 1 you cannot make full use of 20 parts computing resources) or sometimes encounter some exceptions because of the uncertainty of the sharing time and node fault.

2.2 Scalable Problem

Even if the execution can run ideally as scheduled, scalable problem may still exist. Just as shown in Fig. 2(c), the task 1's owner wants to expand his/her task from 8 parts to 12 parts in day 3, which surpasses the computing ability of the system.

Therefore, it is easy to find that a growing-up Voluntary Cloud is hard to cater to customers' various requirements, because it could not provide fault-tolerant and elastic service like some large cloud providers.

Table 1: Task scheduling and execution

	Task 1	Task 2
	sche / exe	**sche / exe**
Day 1	10 / 6 parts	10 / 7 parts
Day 2	7 / 4 parts	10 / 6 parts
Day 3	8 / 8 parts	10 / 8 parts
Day 4	5 / 3 parts	10 / 5 parts
Day 5	/	10 / 4 parts

2.3 System Overview

Service provider should collect some computing resources, accept a series of computing tasks and ensure that these tasks can be com-

pleted on time with collected computing resources. In order to build stable and scalable voluntary-resource cloud infrastructure, an architecture is proposed based on a widely-used p2p structure, Bittorrent [7] just as shown in Fig. 3. Moreover, three tables store the main information of the entire system just like Fig. 4.

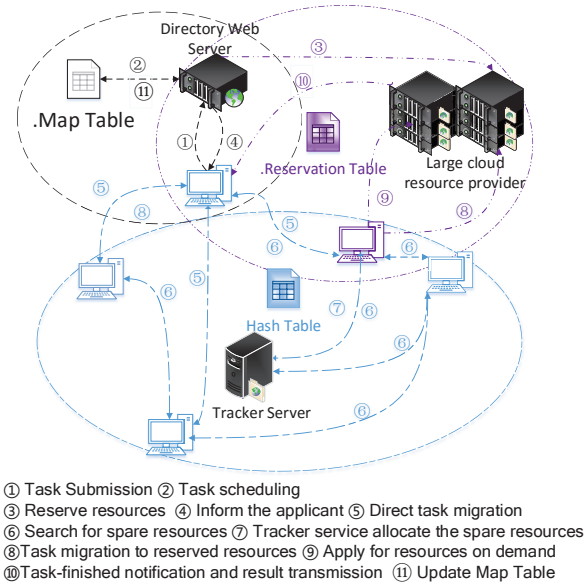


Fig. 3: System Framework

Definition 1. ***(Map Table)*** *It is a table consisting of a set of data elements using a model of vertical columns which are identified by system nodes' id and horizontal rows which stand for each node's multi-attributes which contain the state, CPU, memory, storage, network bandwidth, fault rate and so on.*

In order to define hash table briefly, a relation $\sim$ and a N similar resources closure should be first defined.

Definition 2. ***(Relation $\sim$)*** *It is a relation between an attributes vector $\boldsymbol{a}$ and a requirement vector $\boldsymbol{r}$, where $\boldsymbol{r} = (r_0, r_1, ..., r_m)$ and r_i stands for one requirement or attribute which should be satisfied; $\boldsymbol{a} = (a_0, a_1, ..., a_m)$ represents a part resource's attributes and each a_i is corresponding to r_i; $\boldsymbol{a} \sim \boldsymbol{r}$ means that each attribute a_i in $\boldsymbol{a}$ satisfies each requirement r_i in $\boldsymbol{r}$.*

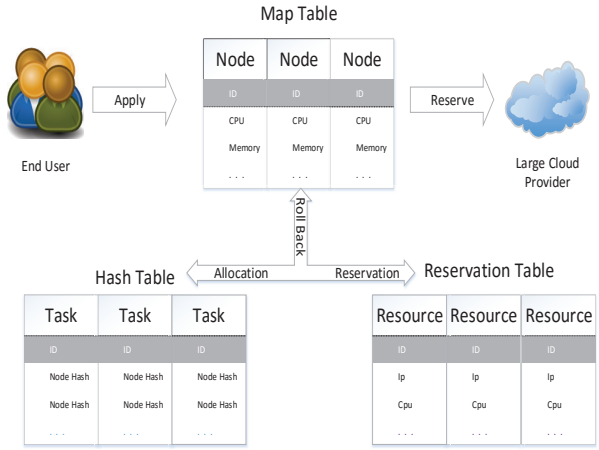


Fig. 4: Three Tables in System Framework

Definition 3. ***(N similar resources closure for requirements $\boldsymbol{r}$ in Set $\boldsymbol{S}$)*** *Given a resources set $\boldsymbol{C} \subset \boldsymbol{S}$, if each attribute vector $\boldsymbol{c} \in \boldsymbol{C}$ satisfies $\boldsymbol{c} \sim \boldsymbol{r}$ and each attribute vector $\boldsymbol{a} \in \boldsymbol{S} - \boldsymbol{C}$ satisfies $\boldsymbol{a} \not\sim \boldsymbol{r}$. The set $\boldsymbol{C}$ is called N similar resources closure for requirements $\boldsymbol{r}$ in Set $\boldsymbol{S}$.*

Definition 4. ***(Hash Table)*** *The hash table records the location of N similar resources which cater to the same task's requirements R in the resource pool S.*

Definition 5. ***(Reservation Table)*** *It is a table to record the location and attributes of reserved resources in some large cloud providers.*

By sending computing resources' information to the web service, the participants join in the Combined Cloud. Fig. 5 shows the entire execution procedure and the detailed steps are specified as follows,

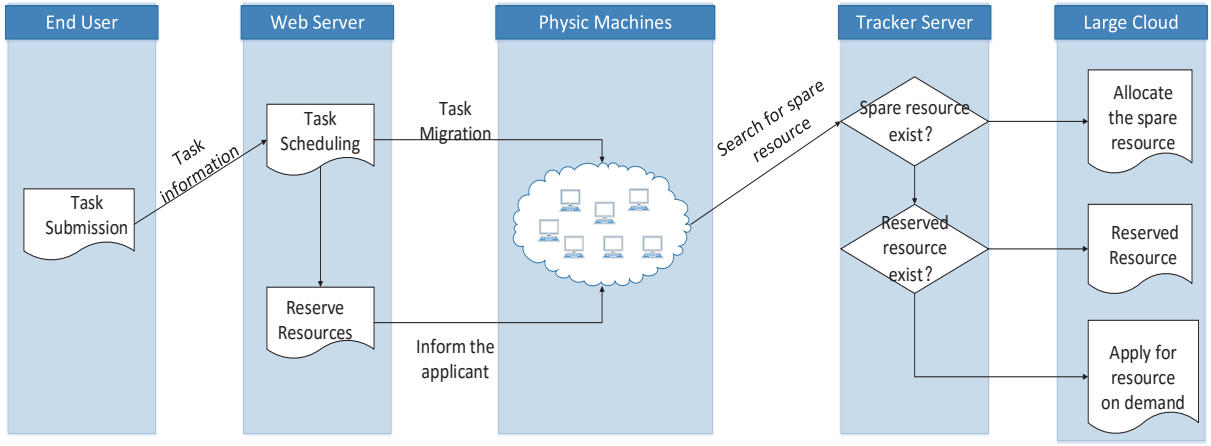


Fig. 5: The entire task execution procedure in the system architecture

(1) **Task Submission:** when a participant applies for a task, he/she needs to submit the task to the web service.

(2) **Task Scheduling:** then the web service searches the Map table, allocates the computing resources and schedules the task if the resources are adequate. Meanwhile, it would record some spare resources which cater to each task's requirements into Hash table.

(3) **Reserve resources:** according to the machines' fault rate recorded in the Map table, the web service reserves a certain amount of resources from Reserved Instance Marketplace and records its location and attributes into Reservation table.

(4) **Inform the applicant:** after the above steps, the web service would inform the applicant of the allocated machine and Reservation table.

(5) **Task migration:** the applicant migrates the task directly.

(6) **Search for spare resources:** under the monitoring of the system, if a task would haven't been executed as scheduled, the node would apply to the tracker server for some spare resources.

(7) **Allocate the spare resources:** the tracker server would search the Hash table and allocate some free nodes to the task.

(8) **Task migration to reserved resources:** if the spare resources in the system are used up, the node which executes the task would migrate a part of the task to the reserved resources according to the Reservation table.

(9) **Apply for resources on demand:** if the reserved resources are used up, the system would apply for new resources on demand.

(10) **Task-finished notification and result transmission:** in the end of execution, the result would be returned to the applicant.

(11) **Update Map Table:** at last, the Map Table containing the state and fault rate would be updated in directory web service and tracker service.

Inherited by Bittorent protocol, this architecture is composed of a centralized web service, a tracker server which monitors all tasks' executions and the nodes' states. The main differences in task execution between this architecture and others based on P2P structure are resource reservation (Step (3)), task migration to the large cloud provider (Step (8)) and fault rate prediction (Step (11)). The task migration has a mature solution in cloud computing [9] and the fault rate could be predicted by some mature regression model.

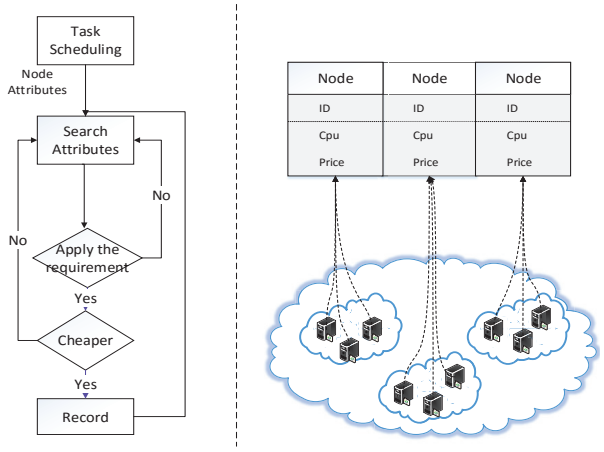


Fig. 6: The procedure of the resource reservation

2.4 Resources Reservation Strategy

The resource query match problem are formulated as follows: Let N be the set of the computing resources in Reserved Instance Marketplace, P be the set of price for it and A be the set of m dimension vector. For each node n_i in N, there is a $\boldsymbol{p_i}$ and a $\boldsymbol{a_i} = (a_{i0}, a_{i1}, ..., a_{im})$ to represent its price and attributes, the optimal resources reservation should be selected from N. In the module of this problem, assume that the resource information of Voluntary Cloud has been collected, which is denoted as matrix [N,P]. The problem is converted to search a part computing resource, which satisfies the requirement and has a minimum price in Matrix $[N, P]$. Fig. 6 represents the procedure from gathering resources to resource query match. Thus, a simple algorithm for resources query match is specified as Algorithm 1.

Algorithm 1 The Procedure of Resource Query Match

Input: $[N, P]$, A^*
Output: the serial number in N, S_n.

```
Sn = 0;
p_min = MAX;
for all n_i ∈ N do
    if n_i ~ A* then
        if P_i < p_min then
            p_min = P_i;
            S_n = i;
        end if
    end if
end for
return S_n;
```

3 Computational Method for Calculating the Quantity of Reserved Resources

A cost-effective, stable and scalable service is beneficial to encourage the development of Voluntary Cloud. Considering the fault rate of each node is predicted by some methods, it would cause an uncertain quantity of fault

nodes. If the reserved resources are less than the real execution need, the supplementary resources would be reserved on demand and else, some money would be wastered on the redundant reserved resources, which damages our economic interests. Thus, a resources reservation strategy is adopted to optimize the system profit in Combined Cloud.

3.1 Problem Formulation

Suppose there are N nodes in Voluntary Cloud denoted by v_i, where $1 \leq i \leq N$. Each node owns $m(v_i)$ MIPS resources, and the fault rate of each node is denoted as $f(v_i)$. The nodes' resources vector and fault rate vector are denoted as $\boldsymbol{m} = (m(v_0), m(v_1), ..., m(v_{N-1}))^{\mathrm{T}}$ and $\boldsymbol{f} = (f(v_0), f(v_1), ..., f(v_{N-1}))^{\mathrm{T}}$. Because the fault rate's prediction exists bias, a function $e(v_i)$ is adopted to indicate the expected prediction error of each node's fault rate. Similarly, the system expected prediction error could be denoted as a vector $\boldsymbol{e} = (e(v_0), e(v_1), ..., e(v_{N-1}))^{\mathrm{T}}$. After defining the resource in the system, we could schedule k tasks, each is denoted as t_i, where $1 \leq i \leq k$. In addition each task needs $b(t_i)$ MIPS resources and the resources requirement vector could be denoted as $\boldsymbol{b} = (b(t_0), b(t_1), ..., b(t_{k-1}))$. Considering that optimizing many tasks at the same time is nontrivial, one task would be used to study and then the result could be extended to the optimization of the entire system. For a task t_i which needs $b(t_i)$ MIPS resources, it would be allocated n nodes, and each is denoted as v_{ij}, where $1 \leq j \leq n$. Resource allocation is denoted as a vector $\boldsymbol{a}_{t_i}$ and $\boldsymbol{a}_{t_i} = (a(v_{i0}), a(v_{i1}), ..., a(v_{i(n-1)}))^{\mathrm{T}}$ where $a(v_{ij})$ stands for the work the node v_{ij} should complete. Besides, each node's fault rate is denoted as a vector $\boldsymbol{f}_{t_i}$, $\boldsymbol{f}_{t_i} = (f(v_{i0}), f(v_{i1}), ..., f(v_{i(n-1)}))^{\mathrm{T}}$ and each fault rate's expected prediction error as $e(v_{ij}) = \epsilon$, which is supposed to be a Gaussian random variable with expectation zero and variance $\hat{\sigma}^2$, that is, $\epsilon \sim N(0, \hat{\sigma}^2)$.

Let P_{r} denotes a MIPS reserved resource's price , P_{d} denotes a MIPS on-demand resource's price and P_{c} denotes the cost of a MIPS resource. Likely, the m MIPS resources are composed of r MIPS reserved, d on-demand and c local resources. Thus, the price and the resource composition need to strictly satisfy the following constraints (1), (2) and (3).

$$P_{\mathrm{r}} \times r + P_{\mathrm{d}} \times d + P_{\mathrm{c}} \times c < P_d \times m, \quad (1)$$

$$d + r + c = b(t_i), \quad (2)$$

$$P_c < P_r < P_d \quad (3)$$

3.2 Resource Reservation Optimization

3.2.1 A task's resource reservation optimization

For a task t_i, after task scheduling and resource allocation, its allocation vector is $\boldsymbol{a}_{t_i}$ and the node fault vector is $\boldsymbol{f}_{t_i}$. Note that due to the existence of deviations of prediction which are to be additive and to satisfy Gaussian distribution, each node's completed work is $a(v_{ij}) \times (f(v_{ij}) + e(v_{ij}))$ and the entire completed work is denoted as $y = \boldsymbol{a}_{t_i}^{\mathrm{T}} \cdot (\boldsymbol{h} - \boldsymbol{f}_{t_i} - \boldsymbol{e}_{t_i})$ where $\boldsymbol{h}$ is a vector $(h_{ij})_{n \times 1}$ with $h_{ij} = 1$. Hence

$$y = \boldsymbol{a}_{t_i}^{\mathrm{T}} \cdot (\boldsymbol{h} - \boldsymbol{f}_{t_i}) - \boldsymbol{a}_{t_i}^{\mathrm{T}} \cdot \boldsymbol{e}_{t_i}, \quad (4)$$

where $\boldsymbol{a}_{t_i}^{\mathrm{T}} \cdot \boldsymbol{e}_{t_i}$ distributes as the sum of multiple Gauss distributions:

$$\boldsymbol{a}_{t_i}^{\mathrm{T}} \cdot \boldsymbol{e}_{t_i} \sim \sum_{j=0}^{n-1} N(0, a^2(v_{ij})\hat{\sigma}^2). \tag{5}$$

Due to the property of Gaussian distribution, it is easy to show that

$$\boldsymbol{a}_{t_i}^{\mathrm{T}} \cdot \boldsymbol{e}_{t_i} \sim N(0, \sum_{j=0}^{n-1} a^2(v_{ij})\hat{\sigma}^2). \tag{6}$$

Without emphasis on the fault rate prediction, a prediction model is needed to ensure that the computational method can calculate the accurate result, which has a little influence on our conclusion. Thus a simple but powerful prediction model, linear model fit by least squares is used to predict the fault rate in Assumption 1.

Assumption 1 (linear model fit by least squares). *It predicts a node's fault rate from the $C \times p$ input matrix $\boldsymbol{X}$, where there are C instances with p dimensions. The number of instances and dimensions satisfies the constraint*

$$C > p \tag{7}$$

In this linear model, one estimates the variance $\hat{\sigma}^2$ by

$$\hat{\sigma}^2 = \frac{1}{C-p-1} \sum_{l=0}^{C-1} (f_l(v_{ij}) - f(v_{ij}))^2 \tag{8}$$

and hence

$$(C-p-1)\hat{\sigma}^2 \sim \sigma^2 \chi^2_{C-p-1}, \tag{9}$$

a chi-squared distribution with $C-p-1$ degrees of freedom. To calculate the true distribution function of $\boldsymbol{a}_{t_i}^{\mathrm{T}} \cdot \boldsymbol{e}_{t_i}$, the standardized coefficient or Z-score could be calculated as

$$z = \frac{\boldsymbol{a}_{t_i}^{\mathrm{T}} \cdot \boldsymbol{e}_{t_i}}{\sqrt{\sum_{j=0}^{n-1} a^2(v_{ij})}\hat{\sigma}} \tag{10}$$

and thus $z \sim t_{C-p-1}$. If $b(t_i) - y - r > 0$, the average cost of $Cost$ can be calculated as

$$E(Cost) = P_{\mathrm{r}} \times r + P_{\mathrm{d}} \times (b(t_i) - y - r) + P_{\mathrm{s}} \times y \tag{11}$$

and else

$$E(Cost) = P_{\mathrm{r}} \times r + P_{\mathrm{s}} \times y. \tag{12}$$

Then, the cost the organizer will pay for a task t_i could be calculated from (11) and (12). Because the $Cost$ is not fixed, the expectation could be calculated as following,

$$\begin{aligned} E(Cost) = \int_{-\infty}^{\delta} [P_{\mathrm{r}} \times r + P_{\mathrm{d}} \times (b(t_i) - y - r) \\ + P_{\mathrm{c}} \times y] dz + \int_{\delta}^{\infty} [P_{\mathrm{r}} \times r + P_{\mathrm{c}} \times y] dz \end{aligned} \tag{13}$$

where

$$\delta = \frac{b(t_i) - r - \boldsymbol{a}_{t_i}^{\mathrm{T}} \cdot \boldsymbol{f}_{t_i}}{\sqrt{\sum_{j=0}^{n-1} a^2(v_{ij})}\hat{\sigma}} \tag{14}$$

δ obeys the $t-distribution$ with the parameter c-p-1, therefore

$$\begin{aligned} E(Cost) = T(\delta)[P_{\mathrm{r}} \times r + P_{\mathrm{d}} \times (b(t_i) - y - r) \\ + P_{\mathrm{c}} \times y] + (1 - T(\delta))[P_{\mathrm{r}} \times r + P_{\mathrm{c}} \times y] \end{aligned} \tag{15}$$

where $T(\delta)$ is the cumulative distribution function of t_{C-p-1}. To calculate the extreme of equation (15), it should be differentiated with respect to r, then

$$\begin{aligned} \frac{dE(Cost)}{dr} = P_{\mathrm{r}} + \frac{dT(\delta)}{d\delta} \times \frac{d\delta}{dr} \cdot P_{\mathrm{d}}(b(t_i) - y - r) \\ - P_{\mathrm{d}} \times T(\delta). \end{aligned} \tag{16}$$

Let $\frac{dE(Cost)}{dr} = 0$, a first order ordinary differential equation should be solved

$$\begin{aligned} \frac{dE(Cost)}{dr} = P_{\mathrm{r}} + \frac{dT(\delta)}{dr} \times \frac{d\delta}{dr} \cdot P_{\mathrm{d}}(b(t_i) - y - r) \\ - P_{\mathrm{d}} \times T(\delta). \end{aligned} \tag{17}$$

Hence, the r can be calculated as,

$$r = b(t_i) - T^{-1}(P_{\mathrm{r}}/P_{\mathrm{d}})\sqrt{\sum_{j=0}^{n-1} a^2(v_{ij})}\hat{\sigma} - \boldsymbol{a}_{t_i}^{\mathrm{T}} \cdot (\boldsymbol{h} - \boldsymbol{f}_{t_i}) \quad (18)$$

From equation (18), it is easy to find that $b(t_i)$ and $\boldsymbol{a}_{t_i}^{\mathrm{T}} \cdot \boldsymbol{f}_{t_i}$ are fixed, so r is correlated with $T^{-1}(P_{\mathrm{r}}/P_{\mathrm{d}})$ and $\sqrt{\sum_{j=0}^{n-1} a^2(v_{ij})}\hat{\sigma}$ which respectively stands for the quantile where $P_{\mathrm{r}}/P_{\mathrm{d}}$ is in the distribution of fault rate and the variance of it. The above analysis indicates that the optimal quantity of the reserved resources is only correlated with the distribution of the fault rate but has nothing with the predictive model. Under the case of one task, the equation (18) is the minimum point that makes our expected cost lowest. But, for the entire system, the situation is much more complex.

3.2.2 The System's Resource Reservation Optimization

In this subsection, the resource reservation optimization algorithm in Algorithm 2 has been proposed. The system's resource reservation optimization is not just linear superposition of each task. There are two different points which are specified as follows. 1) The existence of the spare resource pool in the system: when a task encounters some problems, it could seek some spare nodes from tracker server to complete its work. For an isolated task t_i, after completed, the resources the system donates to it are just correlated with its allocation vector $\boldsymbol{a}_{t_i}$. However, in a system, the resources the system donates to a task are not only correlated with its allocation vector but also the spare resources. 2) the requirements of different tasks are different: for a task, the reserved resources just need to satisfy its requirements. But, the situation changes when you consider the entire system due to diversity of tasks' requirements.

Algorithm 2 Computational method for reserved resources

Input: $[T]$, $[Rp]$, $[Ra]$, N

Output: $[r]$ //The quantity of reserved resources

1: Sort $[T]$ into $[L]$ according to $[T].quantity$
2: Map M, Mr;
3: **for** all $T_i \in [L]$ **do**
4: $M.key = T_i$;
5: $[S] = Scr(N, T_i.require, [Rp])$;
6: $M.value = [S]$;
7: Remove $[S]$ from Rp;
8: **end for**
9: **for** all $T_i \in [L]$ **do**
10: $Mr.key = Scr(1, T_i.require, [Ra])$;
11: $Mr.value \leftarrow Scr(1, T_i.require, [Ra])$;
12: **end for**
13: **for** all Key in Mr **do**
14: List A, F;
15: **for** all T_i in $Mr.value$ **do**
16: $A \leftarrow$ M.getvalue(T_i)
17: **end for**
18: **for** all A_i in A **do**
19: $F_i = f(A_i)$
20: $[r] \leftarrow$ equation (18)
21: **end for**
22: **end for**
23: **return** $[r]$;

A task's N similar resources closure denoted P_{t_i} could be obtained from the Hash Table. If each closure does not overlap, $\forall A_i \in P_{t_i}, A_i \notin P_{t_j}$ where $j \neq i$, then the problems need to be divided into each closure P_{t_i}. There-

fore, the resources allocated by the system in a closure, c, are the sum of allocation vector a_{t_i} and spare resources S.

Algorithm 3 N similar closure for requirements R in set S

Input: N, R, S

Output: $[L]$ //List for N similar resources vector

```
for all S_i ∈ S do
    if S_i ∼ R then [L] ← S_i
    end if
end for
while [L].length > N do
    for all L_i ∈ [L] do
        for all L_j ∈ [L] do
            if L_i ≁ L_j then break;
            end if
            if i = j then
                Remove L_i from [L]; break;
            end if
        end for
    end for
end while
if i ≠ [L].length then Remove L_0 from [L];
end if
return [L];
```

$$c = \boldsymbol{a}_{t_i}^{\mathrm{T}} \cdot (\boldsymbol{h} - \boldsymbol{f}_{t_i}) + \boldsymbol{f}_s \tag{19}$$

It is easy to derive that

$$c = \sum_{i=0}^{x} \boldsymbol{a}_i^{\mathrm{T}} \cdot (\boldsymbol{h} - \boldsymbol{f}_i) \tag{20}$$

where $\boldsymbol{a}_i$ stands for the computing resources in closure P_{t_i} and $\boldsymbol{f}_i$ is the fault rate corresponding to each computing resource. For every closure, the system should select those resources which satisfy the task's requirements and cost us the least money. Due to the positive correlation between the resources' pricing and attributions, the resource should be selected in 1 similar closure for its requirement in set $\{Amazon\ reserved\ resources\}$, denoted by $[R_a]$. Then each closure could be incorporated into a different set S_i according to their 1 similar closure. At last, the quantity of computing resources would be respectively calculated in every set S_i by equation (18). But, if closures overlap with each other, it need to be substituted from the spare sources or be just calculated once in the resources for the lowest price. From algorithmic point, the second method is easier to execute. At first, each task $[T]$ is sorted according to its computing quantity into list $[L]$. Secondly, each task's N similar resources closure is calculated for its requirements R in resource pool $[Rp]$, where the resources vector should be removed if it has been divided into preceding closures. Then, the procedure is same as the method that deals with the closures with no overlap. At last, an algorithm is given to solve N similar closure for requirements R in set S in Algorithm 2, denoted by Scr, and the entire computational method is simplified by Algorithm 3.

4 Performance Evaluation

Two sets of simulations have been conducted to test our system performance and verify our computational method for resource reservation optimization.

4.1 Experimental Setting

To conduct the simulation, the system first randomly generates n participating nodes

Table 2: Parameter Setting

System Parameters Setting		**User Tasks' Demand**	
Parameter	**Value**	**Parameter**	**Value**
number of nodes	2000 $\sim$ 12000	CPU rate	1 $\sim$ 25.6 Gflops
number of processors per node	1, 2, 4, 8	quantity	0.64 $\sim$ 640 Pflo
computation rate per processor	1.2, 2.4, 3.2 Hz	N	N
I/O speed per node	20, 40, 60, 80 MbPS	I/O speed	20 $\sim$ 80 Mbps
memory size per node	512, 1024, 2048, 4096 MB	memory size	512 $\sim$ 4096 MB
disk size per node	20, 60, 120, 240 Gb	disk size	20 $\sim$ 240 Gb
LAN network bandwidth	5 $\sim$ 10 Mbps	N	N
WAN network bandwidth	0.2 $\sim$ 2 Mbps	bandwidth	0.1 $\sim$ 10 Mbps
the mean of fault rate per day	0.05 $\sim$ 0.35	N	N

of which the hardware configuration is randomly selected according to system parameters specified in Table 2. From the table, the *min_capacity* and *max_capacity* could be derived at each resource dimension. For example, along CPU dimension, *min_capacity* and *max_capacity* are $1\times1 = 1$ Gflops and $8\times3.2 = 25.6$ Gflops, respectively which stand for a machine which is only 1 core running at speed of 1 Gflops and 8 cores per node each operating at 3.2 Gflops. Each node's resource price is proportional to its computing capacity, of which the proportional coefficient is λ.

The fault rate in the system is generated from the Gaussian distribution of which u belongs to $[0.05, 0.35]$ and v is equal to 1. When the fault rate is less than zero, it represents that the sharing time of some participants surpasses the scheduling. However, this part of computing resources would be wasted and so the fault rate is equal to zero.

The tasks' requirements are listed in Table 2. The data in Table 2 is collected in Reserved Instance Marketplace in Amazon. In order to reflect the diversity of users' demands, the range of tasks' requirements is relatively large. For instance, the computing quantity of a task in the system ranges from 640 Tflo (tera Floating-point Operations) to 640 Pflo (pera Floating-point Operations). Each tasks' time constraint is set as $86,400$ seconds (one day).

In first experiment, the 100 tasks come into the system from 5 time points in a day. Because the interval of their last time is from 1 to 7, the tasks should be scheduled in time interval $[1, 7]$. Then a greedy algorithm should be used to allocate resources to each task. After task scheduling and resource allocation, the tasks would be implemented respectively on P2P architecture, BFTCloud and Combined

Cloud. In a P2P architecture, the system allocates tasks computing resources without regard to nodes' fault. However, if any node encounters some faults, it would substitute other nodes for it from resource pool. In the BFTCloud architecture, four nodes are set to complete one part computing task and if any node encounters some problems, it would also substitute for it.

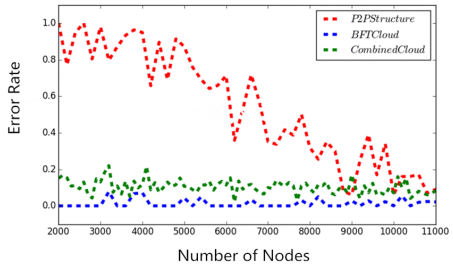


Fig. 7: The fault rate of three architectures

In the second experiment, the part pricing of Amazon EC2 [3] would be adopted in Table 3 to substitute it, where the dph stands for dollar per hour. Then the quantity of the resources the system should reserve would be calculated according to our computational method in Subsection 3.2. At last, the cost in our computational method would be compared with other methods.

4.2 Experimental Result

The computing ability of 8000 nodes could be calculated as $4700.16Pflo$ a day and the average computing task in a day is approximately $4571Pflo$ which is under the computing ability of the 8000 nodes. However, the error rate of it is still more than 0.3, which also results from uncertainty of sharing time.

[3]http://aws.amazon.com/cn/ec2/pricing/ Sep 2016

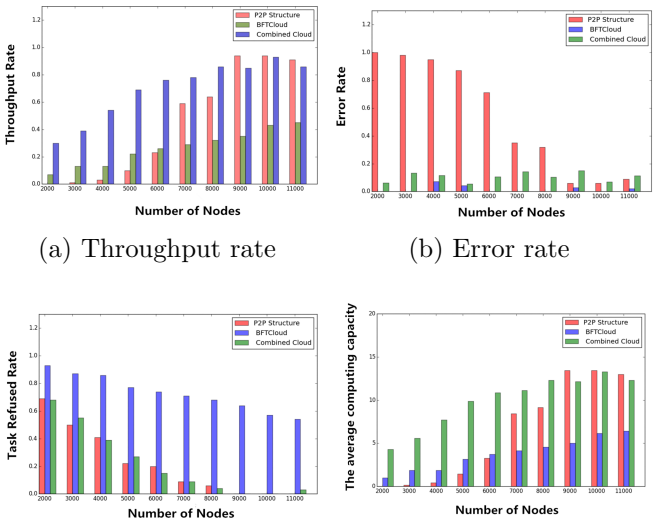
4.2.1 System Performance Experiment

Fig. 7 presents the failed task ratio between Combined Cloud to P2P and BFTCloud architecture with the node number $n = 2000 \sim 11000$. The failed task ratio is defined as the ratio of the number of the tasks which is not finished on time. When $n = 2000$, our model and BFTCloud model are apparently superior to the general P2P structure because there are some reserved resources to replenish some fault nodes. When the number of nodes rises to 6000, the failed ratio in the general P2P structure rapidly descends. In the tail of the curve, the failed ratio in the general P2P structure is close to zero but fluctuates, which results from each node's uncertainty of the sharing time. In our model and BFTCloud, the system fault rate is comparatively low and stable. The throughput rate, error rate, task refused rate and average computing capacity of these three structures are respectively presented in Fig. 8. From Fig. 8(a), (c) and (d), the throughput rate, the task refused rate and the average computing capacity of the P2P structure and Combined Cloud are approximately equal when the number of nodes exceeds 9000. However, when the nodes are under 5000, the fluctuation and under-capacity of the P2P structure are reflected. To BFTCloud, the low error rate is embodied in Fig. 8(b) but the low throughput, high task refused rate and low computing capacity are meanwhile reflected in Fig. 8(a) (c) and (d) respectively, which results in low economic benefit in the system. Thus, Combined Cloud is more stable than the general P2P structure and possesses better system per-

Table 3: The part pricing of Amazon EC2 Instances

Name	vCPU	ECU	Memory	Storage	On-demand Pricing	Reserve Pricing
m3.medium	1	3	3.75	4 SSD	0.067 dph	0.0403 dph
m3.large	2	6.5	7.5	32 SSD	0.133 dph	0.0814 dph
m3.xlarge	4	13	15	80 SSD	0.266 dph	0.1631 dph
m3.2xlarge	8	26	30	160 SSD	0.532 dph	0.3243 dph

formance than BFTCloud.



(a) Throughput rate (b) Error rate

(c) Task refused rate (d) The average computing capacity

Fig. 8: The performance of three architectures

4.2.2 Calculate Method Experiment

In Fig. 9, 10 times comparison experiments were conducted to respectively evaluate our computational method in 2000 $\sim$ 8000 nodes. The red curve represents the cost if the system purchases the reserved computing resources at demand price and the blue is the cost at reserved price. The green curve is the system cost with our computational method. It is easy to find that the green curve is under the red curve which represents that the cost in our system is at least superior to the cost at on-demand price.

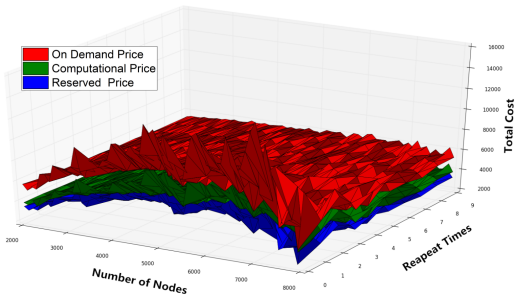


Fig. 9: The costs of three architecture

Furthermore, when the number of the nodes is under 6000, the green curve approaches the blue curve which implies that our computational cost is almost equal to the lowest cost. When the number of the nodes is beyond 6000 and the experiment times are under 5 times, the red curve fluctuates largely but the green curve is relatively stable which demonstrates the cost in our system fluctuates lowly. At last, when experiment is repeated many times, the green curve is approaching the reserved approach which represents that the cost tends to a stable and economically opti-

mized cost.

5 Related Work

With the development of computer hardware, our personal computers possess a powerful computing ability. More and more researches try to organize these computing resources to provide computing services to the public. Voluntary Cloud [5][7][10], Self-organizing Cloud [11][12] and Collaborative Computing Cloud [13][14][15][16] are just three new paradigms for cloud computing. Compared with traditional cloud computing or Grid computing, it shows unique characteristics such as small scale, instable and low performance. Thus, the researchers pay their attention to the following issues in these novel cloud paradigms:

1) The self-organizing model and system architecture: a new resource management method [17] for Cloud Computing, resource self-organizing model, is proposed that forms self-government resource groups in the absence of centralized management control and dynamically optimizes the organizational structure of resources in accordance with resource changes.

2) The resource allocation and task scheduling in cloud computing [18][19][20]: the Voluntary Cloud is different from the traditional Grid model in the consumption manner. Grids generally assume exclusive resource usage to ensure users' QoS. Thus, some approaches [5][21][11] are proposed to not only achieve the maximum resource utilization using the proportional share model (PSM), but also deliver provably and adaptively optimal execution efficiency. In addition, the resource search and match problem in traditional P2P structure similarly exist in Voluntary Cloud. The multi-dimensional range search problem is known to be challenging as contentions along multiple dimensions could happen in the presence of the uncoordinated analogous queries. Moreover, network delay will affect your matching rate. Accordingly, a few researches [22] design a novel resource discovery protocol to do with many analogous queries and obtain a high matching rate.

3) The security and privacy problem in cloud computing [23][24]: in Voluntary Cloud, each computing node is distributed in different places and belongs to different participants, which results in many security and privacy problems. A novel approach [25] tackles both loss of trust and security control problems by enabling cloud consumers to extend their security management process to include cloud hosted assets in collaboration-based cloud computing environment. Moreover, data deduplication is one of important data compression techniques for eliminating duplicate copies of repeating data, and has been widely used in cloud storage. Therefore, to better protect data security, some researchers [26] make the attempt to formally address the problem of authorized data deduplication. Besides, the study on the fairness between the nodes in clouds is a new direction in resource allocation. In [18], it proposes a performance centric fairness resource allocation method to solve this problem.

4) Pricing mechanism: the Voluntary Cloud or Self-organizing Cloud is sometimes looked as a market where users share and trade resources. In this market, the resource pricing is changing all the time and users need to purchase different types of resources from one or

more resource providers to cater to their personal demand. Thus, a few researches [27] are devoted to designing a dynamic resource pricing scheme suitable for every rational end user.

To the best of our knowledge, almost all the work about Voluntary Cloud is proposed with assumption that the entire system has been well established and possessed sufficient resources. However, it is unrealistic for a long time, because Voluntary Cloud is a new paradigm which is hard to be received by the public with a short time. Thus, our work is devoted to designing a system architecture which is more effective than other system architectures [7][10] when a Voluntary Cloud is being organized or in a small scale. Moreover, a computational method has been designed to help the organizer who wants to adopt our architecture to make more profit. Our computational method is a heuristic method with $\mathcal{O}(n)$ time complexity to solve our problem. Because the problem is unique and local that there is not related work about it. However, the experiment results demonstrate our method is effective and stable.

6 Conclusion and Future Work

Due to the development of Voluntary Cloud, many problems are emerging beyond our expectation. A lot of researchers assume that the resource pool should be considerable but it is unrealistic in reality. However, in small scale cloud, resource pool is not sufficient which is an issue that damages the system stability and elasticity. This paper proposed a novel architecture that the system can reserve or request resources from some large cloud providers. It ensures that each task is completed on time and is just a compromised solution to help Voluntary Cloud transit itself from small scale cloud to an independent and autonomous cloud. In addition, a computational method is proposed to optimize our economic benefit in resources reservation step. At last, two sets of simulations are conducted to validate the effectiveness of our system framework and resource reservation strategy.

In the future, a decentralized architecture could be conducted by adopting DHT (distributed hash table) method which is more complex to design but more convenient to organize. In addition, the incentive mechanism is a novel but important research direction. How to encourage the participants to donate more computing resources and attract more people to participate in Voluntary Cloud is a key issue for the organizer. Thus, some tools such as game theory could be used to design an incentive mechanism for our system. It is a good way to attract more people to participate in our system and encourage the participants to contribute more computing resources, which would make the Combined Cloud more effective and stable.

At last, the rise of Voluntary Cloud may be harmful to some large cloud providers. It would scramble for customers and computing resources. Moreover, it would prevent the Voluntary Cloud from reserving or requesting resources from them. In this case, the strategy presented in this paper should be improved. The game theory method could be used to analyze this situation and achieve a stable state solution. In this stable state solution, both the large cloud providers and Voluntary Cloud

providers would not change their pricing methods. Accordingly, it forms a Nash Equilibrium between the Combined Cloud and large cloud providers.

Acknowledgment

This paper is partially supported by the National Science Foundation of China under Grant No. 91318301 and No. 61672276, the Key Research and Development Project of Jiangsu Province under Grant No. BE2015154, BE2016120, the Collaborative Innovation Center of Novel Software Technology, Nanjing University and the EU FP7 CROWN project under grant number PIRSES-GA-2013-610524.

References

[1] Butt S, Lagar-Cavilla H. Self-service cloud computing. In *Proc. ACM Conference on Computer and Communications Security*, July 2012, pp. 253–264.

[2] Khalaf M N, Al-Ghuwairi A R, Al-Yasen L. A trust framework for ranking user as a cloud provider in peer-to-peer cloud system. In *Proc. the International Conference on Internet of things and Cloud Computing*, June 2016, pp. 391–416.

[3] Shao J, Lu R. Fine: A fine-grained privacy-preserving location-based service framework for mobile devices. In *Proc. IEEE International Conference on Computer Communications*, January 2014, pp. 244–252.

[4] Zhu C, Leung V C M. Trust assistance in sensor-cloud. In *Proc. Computer Communications Workshops*, June 2015, pp. 342–347.

[5] Wang H, Wang F, Liu J, Groen J. Measurement and utilization of customer-provided resources for cloud computing. In *Proc. IEEE International Conference on Computer Communications*, December 2012, pp. 442–450.

[6] Foster I, Zhao Y. Cloud computing and grid computing 360-degree compared. *Grid Computing Environments Workshop Gce*, 2009, 5: 1–10.

[7] Zhang Y, Zheng Z, Lyu M R. Bftcloud: A byzantine fault tolerance framework for voluntary-resource cloud computing. In *Proc. Cloud Computing*, June 2011, pp. 444–451.

[8] Yao M, Lin C. An online mechanism for dynamic instance allocation in reserved instance marketplace. In *Proc. Computer Communication and Networks*, September 2014, pp. 464–571.

[9] Zhang W, Tan S. A survey on decision making for task migration in mobile cloud environments. *Personal and Ubiquitous Computing*, 2016, 20(3): 295–309.

[10] Ryden M, Oh K. Nebula: Distributed edge cloud for data intensive computing. In *Proc. Cloud Engineering*, March 2014, pp. 491–492.

[11] Di S, Wang C L. Dynamic optimization of multiattribute resource allocation in self-organizing clouds. *IEEE Transactions on Parallel and Distributed Systems*, 2013, 24(3): 464–478.

[12] Megahed M, Ismail R M. An enhanced cloud-based view materialization approach for peer-to-peer architecture. In *Proc. Advanced Machine Learning Technologies and Applications*, July 2012, pp. 491–492.

[13] Ragan-Kelley B, Walters W A. Collaborative cloud-enabled tools allow rapid, reproducible biological insights. *Isme Journal*, 2013, 7(3): 461–464.

[14] Valilai O F, Houshmand M. A collaborative and integrated platform to support distributed manufacturing system using a service-oriented approach based on cloud computing paradigm. *Robotics and computer-integrated manufacturing*, 2013, 29(1): 110–127.

[15] Karnouskos S, Colombo A W. A soa-based architecture for empowering future collaborative cloud-based industrial automation. In *Proc. Annual Conference on IEEE Industrial Electronics Society*, May 2012, pp. 5766–5772.

[16] Liu G, Shen H. An efficient and trustworthy resource sharing platform for collaborative cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, 2014, 25(4): 862–875.

[17] Lin W, Qi D. Research on resource self-organizing model for cloud computing. In *Proc. Internet Technology and Applications*, June 2010, pp. 291–316.

[18] Chen L, Feng Y. Towards performance-centric fairness in datacenter networks. In *Proc. IEEE International Conference on Computer Communications*, January 2014, pp. 1599–1607.

[19] Kayed A, Akijian T. Resource allocation technique to obtain energy efficient cloud. In *Proc. The International Conference on Engineering and Mis*, March 2015, pp. 722–741.

[20] Liao X, Liu C, Mccoy D. Characterizing long-tail seo spam on cloud web hosting services. In *Proc. International Conference on World Wide Web*, March 2016, pp. 587–601.

[21] Wang H, Wang F. Resource provisioning on customer-provided clouds: Optimization of service availability. In *Proc. IEEE International Conference on Communications*, December 2013, pp. 2954–2958.

[22] Di S, Wang C L. Probabilistic best-fit multi-dimensional range query in self-organizing cloud. In *Proc. International Conference on Parallel Processing*, September 2011, pp. 763–772.

[23] Al-Qurishi M, Al-Rakhami M, Alrubaian M, Alamri A. A framework of knowledge management as a service over cloud computing platform. In *Proc. International Conference on Intelligent Information Processing, Security and Advanced Communication*, May 2015, pp. 322–331.

[24] Wang Z, Chen H. Fault tolerant barrier coverage for wireless sensor networks. In *Proc. IEEE International Conference on Computer Communications*, January 2014, pp. 1869–1877.

[25] Almorsy M, Grundy J. Collaboration-based cloud computing security management framework. In *Proc. Cloud Computing*, September 2011, pp. 364–371.

[26] Li J, Li Y K. A hybrid cloud approach for secure authorized deduplication. *IEEE Transactions on Parallel and Distributed Systems*, 2015, 26(5): 1206–1216.

[27] Mihailescu M, Teo Y M. Dynamic resource pricing on federated clouds. In *Proc. IEEE International Symposium on Cluster Computing and the Grid*, May 2010, pp. 513 – 517.