Three-side Gaming Model for Resource Co-allocation in Grid Computing

Peng Xiao, Dongbo Liu, Xilong Qu
School of Computer and Communication, Hunan Institute of Engineering, Xiangtan, 411104, China
Email: xpeng4623@yahoo.com.cn

Abstract—Co-allocation is a fundamental infrastructure to aggregate heterogeneous and distributed resources in grid environments. Although it has been studied extensively, co-allocation under the constraints to budget and deadline still remains an opening issue, which means that tradeoff between user QoS requirements and system performance should be agreed. In this paper, a novel agent-based two-phase co-allocation is proposed, which optimizes resources deployment and price scheme through a two-phase co-allocation mechanism, and applies queuing system to model the working of resources for providing quantitative guarantee for application’s deadline requirement. Extensive simulations are conducted to evaluate the effectiveness and performance of the model by comparing with other three co-allocation policies in terms of deadline violation rate, resource benefits and utilization. Experimental results show that the two-phase model can significantly improve the QoS satisfaction for those grid applications with constraints to budget and deadline.

Index Terms—grid computing, QoS guarantee, deadline, computing economy, gaming theory

I. INTRODUCTION

Grid computing [1] has emerged as the next generation of parallel and distributed platform that aggregates dispersed heterogeneous resources for solving high-end applications, which frequently require access to multiple resources across different sites. Therefore, resource co-allocation becomes an important issue with increasing attention [2]. In computational Grid, co-allocation is generally performed by meta-scheduler when a job’s resource demands beyond the capacity of any single site. As an effective technique, advance reservation has been widely used to provide QoS guarantees for co-allocating resource across multiple sites [3]. However, advance reservation can only ensure the availability of resources at the required times [4, 7], but cannot provide guarantees for other QoS requirements of applications, i.e. budget and deadline.

In this work, we focus on the QoS-based co-allocation in computational Grid. Our goal is to design an effective co-allocation model, which can provide reliable QoS guarantee for those applications with constraints to budget and deadline. To do this, we introduce a novel concept (namely Virtual Resource Agent) into co-allocation and design a two-phase co-allocation model. In the model, Virtual Resource Agents optimize resources deployment and price scheme through a two-phase co-allocation mechanism, and apply queuing system to model the working of resources for providing quantitative guarantee for grid application’s deadline requirement. In this way, heterogeneous computing resources can be organized in an efficient way to meet application’s budget and deadline requirements.

The rest of this paper is organized as follows. Section II presents the related work. In section III, we introduce the two-phase co-allocation model. In section IV, we analyze the model theoretically. In section V, simulations are conducted to verify the effectiveness and performance of the proposed model. Finally, Section VI concludes the paper with a brief discussion of future work.

II. RELATED WORK

Since co-allocation has been a fundamental infrastructure in Grid resource management and task scheduling, many co-allocation architectures have been developed. In Legion [5], co-allocation is supported by an entity called as Enactor, which relies on advance reservation to allocate multiple resources. In Globus [6], GARA [2-3] has been developed for providing atomic and interactive co-allocation strategies. In [8], Waldrich et al. develop a meta-scheduler implementation, which relies on negotiating with local scheduler to determine a common time slot where all required resources are available for the starting time of applications.

Besides above architectures, many co-allocation models and policies are proposed to optimize certain performance metrics, i.e. mean response time, resource utilization, load balance and etc. In [9], Leinberger et al. propose two backfilling-based heuristics (FCFS/BB and FCFS/BL) for K-resource co-allocation. Their simulations show that load-balancing-based co-allocation policy outperforms classical policy such as FCFS/FF over 50% in terms of mean response time. In [10], Mohamed et al. propose the Close-to-Files co-allocation policy, which tries to place jobs on clusters that are close to the input files so as to reduce communication overhead. To evaluate the performance of various co-allocation policies, Bucur and Epema [11-13] conduct extensive experiments in large-scale Grid testbed DAS-2[14]. Based on their experimental results, they draw an conclusion that workload-aware co-allocation policies are effective to reduce the mean response time and obtain better load-balance.
Unfortunately, few of above studies has addressed the issue of resource co-allocation for Grid applications under the constraints of budget and deadline at the same time. Nimrod-G [15] is a famous grid system that uses computing economy driven architecture for managing resources and scheduling task. In Nimrod-G, three adaptive algorithms for deadline and budget constricted scheduling are proposed [16]: Cost Optimization, Time Optimization, and Conservative Time Optimization. However, the implementations of the three algorithms do not provide any quantitative deadline guarantee for applications when the workload on resources changes dynamically.

Currently, game theory has been widely used to solve the resource allocation problem in Grid computing[17-19]. Many studies assume that participants in games are selfish, and then propose many methods to find the equilibrium solution of resource price or allocation scheme. In [19], Khan classifies resource allocation models as cooperative, semi-cooperative and non-cooperative. By extensive simulations, Khan indicates that agent-based cooperation model is effective for resource allocation. However, Khan’s cooperative model is of very high computational complexity, which inspires us to find more efficient methods.

III. TWO-PHASE CO-ALLOCATION MODEL

The system model considered in this work is shown in Fig. 1, which is based on the conventional multicluster computational Grid model that described in [20]. It consists of several Computing Elements (CE) each representing a homogeneous cluster, a set of Virtual Resource Agents (VRA), and a meta-scheduler.

In this model, meta-scheduler works as follows: when a job arrives, it selects suitable VRAs based on the job’s budgets and deadline, then dispatches the job to those selected VRAs. The VRAs work as follows: all the VRAs buy resources from the system at a uniform price \( p^* \) (details about “uniform price” can be seen in section IV) through the resource price negotiator component, and then sell them to clients at different retail prices. The VRAs can change their size dynamically at runtime by adjusting their resource quantity \( \langle c_1, c_2, \ldots, c_n \rangle \) as shown in fig. 1. The reasons that we introduce VRAs to the system are two-fold: firstly, it provides a reasonable resource price scheme to meet job’s budget constraint as well as improve resource benefits; secondly, it helps us modeling the working of resources so as to provide quantitative guarantee for job’s deadline constraint.

In this two-phase co-allocation model there are three types of participants: the system, VRAs and clients. The system and VRAs cooperate with each other, as they both aim at maximizing resources utilization and system benefits on behalf of resource providers. On the other hand, the relationship between the VRAs and the clients is non-cooperative, as the clients hope to minimize their costs, which would inevitably lower down the benefits of resource providers. According the above description, it can be seen that the three types of participants form a three-site allocation model that similar to Producer-Retailer-Client model [21], in which co-allocation is separated into two phases. In the following work, we will present the validity and solution of this two-phase co-allocation model in theory, and devise a VRA-based co-allocation policy.

IV. ANALYSIS AND SOLUTIONS OF TWO-PHASE MODEL

A. Utility Functions

We first give the utility functions of the three types of participants respectively. In next sections, we will analyze the two-phase co-allocation model based on these utility functions. As the system sells its computing resources to VRAs at a uniform price \( p^* \), its utility function can be simply defined as

\[
U^S = p^* \cdot \sum_{c_e} s_i
\]

where \( s_i \) is the number of computing resources in \( c_e \).

Here, we deliberately ignore the various prices of different computing resources, and let VRAs to make the decision of price scheme. From the view of a whole, the system only needs to care about the total benefits, which can be easily tuned by adjusting \( p^* \). This strategy is inspired by Producer-Retailer-Client model.

Let \( c_i \) be the number of computing resources in \( V_i \), \( \rho_i \) be the resource utilization rate of \( V_i \). Thus, the utility function of \( V_i \) can be defined as

\[
U^i = c_i \cdot p^* \cdot \rho_i - p^* \cdot c_i
\]

where \( p^* \) is the resource price set by \( V_i \) for clients. Besides the utility function of individual VRA, we also care about the total benefits of all the VRAs. So, we define the total utility function of all VRAs as following

\[
U^T = \sum_{c_i} (c_i \cdot p^* \cdot \rho_i) - p^* \cdot \sum_{c_i} c_i
\]

For a job \( j \), we characterize it by a 3-tuple: \( \langle r_j, b_j, d_j \rangle \), where \( r_j \) is the resource demands, \( b_j \) is the budgets, \( d_j \) refers to deadline. Let \( J \) be the set of
VRAs being allocated to job \( j \), and \( r^*_j \) be the amount of resources allocated from \( V_j \) to execute job \( j \), then the cost of job \( j \) is \( \sum_{\nu \in \nu}(r^*_j \cdot p_\nu) \). As the guarantee of deadline is not a quantitative measure, we map it as a probability. Let \( E_i \) be a random event representing \( V_i (i \in J) \) can meet the deadline of job \( j \), then the probability that the deadline of job \( j \) can be satisfied is expressed as \( \prod_{\nu \in \nu}\Pr(E_i) \). So we define the utility function of job \( j \) as follows

\[
U^*_j = \prod_{\nu \in \nu}\Pr(E_i) / \sum_{\nu}(r^*_j \cdot p_\nu) \tag{4}
\]

**B. Solution of Cooperative Gaming Model**

The VRAs and the system both represent the benefits of resource providers that wish to maximize resources utilization and the system benefits, so we use cooperative model to describe their relationship. In this cooperative model, a solution pair \( < p^*,C > \) will be derived, where \( C=\{c_1,c_2,...,c_n\} \) is a vector representing the resource quantity in each VRA.

Given the current price set by the system is \( p^* \), by using (2) and (3) we can obtain the VRAs’ benefits \( (U^*_{V_1},U^*_{V_2},...,U^*_{V_n}) \), and the total VRAs benefits \( U^{V^*} \). If \( U^{V^*} > 0 \), then it means that the current price \( p^* \) is too low. As mentioned above, the relationship between the system and VRAs is cooperative, so we consider the benefits obtained by VRAs as the system’s benefits. In this way, the system can set a new price \( p_1^*=(U^*_{V_1}+U^*_{V_2}+...+U^*_{V_n})/\sum_{\nu\in\nu}q^*_\nu \) to resources. It is clear that the new price \( p_1^* \) will not affect the whole benefits of system.

Under the new price \( p_1^* \), we can get a new VRAs’ benefits vector, denoted as \( (U^*_{V_1},U^*_{V_2},...,U^*_{V_n}) \). From the definition of \( U^*_{V_i} \), we can known that if \( U^*_{V_i} > 0 \) then \( \forall i \in \{1...n\} \) \( U^*_{V_i} > U^*_{V^*} \), which means increasing \( p^* \) to \( p_1^* \) will decrease the benefits for all VRAs. Thus, there are three cases we should consider:

- \( U^*_{V^*} > 0 \) and \( U^*_{V_i} > 0 \): In this case, the benefits of \( V_i \) is still positive even it had to pay a higher price for resources. It is suggested that more resources should be allocated to \( V_i \).
- \( U^*_{V^*} > 0 \) and \( U^*_{V_i} < 0 \): In this case, the \( V_i \) can not get benefits under the new price \( p_1^* \). So allocating more resource is not useful to increase the system benefits.
- \( U^*_{V^*} < 0 \) and \( U^*_{V_i} < 0 \): It is suggested that resources in \( V_i \) should be shrunk to decrease the benefits losing.

Based on the above analysis, we can get a new solution pair \( < p_1^*,C > \), where \( p_1^* \) is the new resource price decided by the system, \( C=\{c_1,c_2,...,c_n\} \) is the vector representing the new resource quantity of each VRA. As to the case \( U^*_{V^*} < 0 \), the analysis is similar to the case \( U^*_{V^*} > 0 \), so we skip it for simplicity. The algorithm 1 is to obtain \( < p^*,(c_1,...,c_n) > \), in which \( S^*(p^*)=\{V_i | U^*_{V_i} > 0\} \) is the set of VRAs with positive benefits at \( p^* \); \( S^-(p^*)=\{V_i | U^*_{V_i} < 0\} \) is the set of VRAs with negative benefits; \( S^0(p^*)=\{V_i | U^*_{V_i} = 0\} \) is the set with zero benefits.

**Algorithm 1:** Obtaing \( < p^*,(c_1,...,c_n) > \) as cooperative gaming model solution

**Input:** \( < p^*,(c_1,...,c_n) > \)

**Output:** \( < p_1^*,(c_1,...,c_n) > \)

**Begin**

1. \( p_1^* = \frac{1}{n} \sum_{\nu\in\nu}(p_\nu \cdot p_\nu) \)
2. for \( i = 1 \) to \( n \)
3. calculate \( U^*_{V_i} \) at the new price \( p_1^* \)
4. if \( V_i \in S^-(p_1^*) \) then
5. add \( i \) to \( \text{expand\_list} \)
6. else if \( V_i \in S^+(p_1^*) \) then
7. add \( i \) to \( \text{shrink\_list} \)
8. end if
9. end for
10. for each \( i \) in \( \text{shrink\_list} \)
11. \( c_i = (1-\Delta k) \cdot c_i \)
12. find \( j \) in \( \text{expand\_list} \), which satisfying one of the two conditions: (1) the amount of resources come from \( CE_i \), in \( V_j \) is maximal; (2) \( U^*_{V_j} \) is maximal in the VRAs that need to be expanded.
13. \( c_j = c_j + \Delta k \cdot c_i \)
14. end for
**End**

**C. Solution of Non-cooperative Gaming Model**

The meta-scheduler assigns jobs to VRAs based on client utility function. The clients tend to select the VRAs with lower retail price and higher resource quantity, because those VRAs are more likely to be able to meet job’s budget and deadline constraints. Although a high value of retail price might bring about better benefits for the VRA, its resource utilization rate will be lowered too. On the other side, a low retail price can lead to high resource utilization rate, however, if the benefits of the VRA become negative, its resource quantity will be reduced in the next adjustment of \( < p^*,(c_1,...,c_n) > \). So, the solution of the no-cooperative model between VRAs and client jobs are the resource retail prices vector \( (p_1,...,p_n) \).

According to computing economy, if the resource utilization rate is in high level, a VRA can reduce its retail price, yet still maintain its benefits in a relative high level. So we consider the retail price is a decreasing function of resource utilization rate, denoted as \( p_i(r_\nu) \). Then, the utility function of \( V_i \) can be rewritten as follows

\[
U^*_{V_i} = c_i \cdot p_i(r_\nu) \cdot p_\nu - p^* \cdot c_i \tag{5}
\]

Let \( \frac{dU^*_{V_i}}{dp_\nu} = 0 \), we can get the equation (6). Denote the solution of equation (6) as \( \rho^*_\nu \). It is clear that the maximal
value of $U^j$ can be obtained when $\rho = \rho^*_j$. So we call $\rho^*_j$ as the optimal resource utilization rate of $V_j$.

$$\rho^*(\rho) \cdot \rho + p_r(\rho) = 0 \quad (6)$$

However, $V_j$ cannot set its $\rho$ as $\rho^*_j$ by itself. Instead, $V_j$ can only change its retail price to influence the clients’ resource selection so as to optimize its benefits. By comparing the difference between $\rho^*_j$ and $\rho$, a VRA can decide whether increasing or decreasing its retail price. The process is as follows: if $\rho < \rho^*_j$, the $V_j$ would decrease its price, else the $V_j$ would increase its price. This process will be performed repeatedly until an optimal retail price scheme is achieved.

Finally, we should figure out the probability that the deadline constraint of a job can be guaranteed, which is shown in formulae (4) and expressed as $\prod_i \Pr(\omega_i)$. We assume that the arrival of jobs in $V_j$ is a Poisson process with rate $\lambda_j$, and the execution time of jobs follows Exponential distribution with rate $\mu$. Therefore, a VRA can be modeled as a $M/M/C$ queueing system [22]. So, the utilization rate of $V_j$ can be expressed as $\rho = \lambda_j(c_j \cdot \mu)$. In this paper, we only consider the case $\rho < 1$.

**Theorem 1.** If VRA is modeled as $M/M/C$, queueing system, then the probability that $V_j$ can guarantee a job’s deadline is

$$\Pr(\omega_i) = \prod_i \Pr(\omega_i)$$

where $\delta = \left[ \sum_{n=0}^{c_j} \frac{(\rho \cdot c_j)^n}{n!} \cdot \frac{1}{c_j \cdot (1 - \rho)} \right]^{-1}$ and $d_j$ is the relative deadline relative to its arrival time.

**Proof:** Let $\psi_i$ be a random variable representing the number of waiting jobs in $V_j$. According to queuing theory, the probability that there are $k$ waiting jobs in $V_j$ is

$$\Pr(\psi_i = k) = \delta \cdot \frac{\rho^k \cdot c_j}{c_j!} \cdot \frac{(\rho \cdot c_j)^n}{n!}, \quad k > 0 \quad (7)$$

$$\Pr(\psi_i = 0) = \sum_{n=0}^{c_j} \frac{(\rho \cdot c_j)^n}{n!} \cdot \frac{1}{c_j \cdot (1 - \rho)}, \quad k = 0 \quad (8)$$

Let $\omega_i$ be a random variable representing the completion time (including waiting time and execution time) of a job in $V_j$, then the probability that $V_j$ can guarantee the job’s deadline $d_j$ is expressed as

$$\Pr(\omega_i \leq d_j) = \Pr(\omega_i \leq d_j) \quad (9)$$

For $M/M/C$ queueing system, the service rate is $c_j \mu$, which means the system can complete $c_j \mu$ jobs in a unit time. So, the amount of jobs that $V_j$ can complete in period $d_j$ is $c_j \mu d_j$. Therefore, the probability that $V_j$ can guarantee a job’s deadline $d_j$ is equal to the probability that the waiting jobs in $V_j$ is not more than $c_j \mu d_j - 1$.

By (7)-(8)(9), we can get that

$$\Pr(\omega_i \leq d_j) = \Pr(\psi_i \leq c_j \cdot \mu \cdot d_j - 1)$$

$$= \sum_{k=0}^{\infty} \Pr(\psi_i = k)$$

$$= \sum_{n=0}^{c_j} \delta \cdot \frac{\rho^k \cdot c_j}{c_j!} \cdot \frac{(\rho \cdot c_j)^n}{n!} \cdot \frac{1}{c_j \cdot (1 - \rho)}$$

According to the theorem 1, we can calculate the deadline guarantee for a single task when dispatching it onto certain resources. In this paper, we use this deadline guarantee to define the client’s utility function so as to reflect the QoS requirement of real-time applications. Therefore, the formulae (4) can be rewritten as following

$$U^j = \prod_i \Pr(\omega_i \leq d_j) \quad (10)$$

The meta-scheduler selects the best VRAs to execute client jobs based on formulae (10). It is noteworthy that client utility function consists of two parts: cost and deadline guarantee. In practice system, the meta-scheduler may choose to optimize cost, or deadline guarantee, or the ratio of two. In simulations, we choose the last policy as our VRA-based co-allocation policy to evaluate the performance of the model. In this paper, we use the M/M/C queue model to describe the working of grid resources as shown in Theorem 1. Other types of queue model can also be applied to calculate the deadline guarantee, such as M/M/1, G/M/1 and etc. We only present the approach based on M/M/C and omit the others due to the limitation of space.

V. EXPERIMENTS AND PERFORMANCE EVALUATION

A. Experimental Settings

We use GridSim [23], a distributed resource management and scheduling simulator, to evaluate the performance of the proposed model. A multi-cluster computational Grid model is constructed, which consists of ten clusters. The detail setting of each cluster is listed in Table 1.

<table>
<thead>
<tr>
<th>TABLE 1</th>
<th>GRID SETTINGS IN SIMULATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Processor number</td>
</tr>
<tr>
<td>CE_1</td>
<td>70</td>
</tr>
<tr>
<td>CE_2</td>
<td>120</td>
</tr>
<tr>
<td>CE_3</td>
<td>150</td>
</tr>
<tr>
<td>CE_4</td>
<td>200</td>
</tr>
<tr>
<td>CE_5</td>
<td>110</td>
</tr>
<tr>
<td>CE_6</td>
<td>50</td>
</tr>
<tr>
<td>CE_7</td>
<td>120</td>
</tr>
<tr>
<td>CE_8</td>
<td>80</td>
</tr>
<tr>
<td>CE_9</td>
<td>300</td>
</tr>
<tr>
<td>CE_10</td>
<td>160</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>MIPS</th>
<th>Price / MIPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE_1</td>
<td>377</td>
<td>12</td>
</tr>
<tr>
<td>CE_2</td>
<td>410</td>
<td>22</td>
</tr>
<tr>
<td>CE_3</td>
<td>380</td>
<td>35</td>
</tr>
<tr>
<td>CE_4</td>
<td>285</td>
<td>17</td>
</tr>
<tr>
<td>CE_5</td>
<td>285</td>
<td>17</td>
</tr>
<tr>
<td>CE_6</td>
<td>515</td>
<td>25</td>
</tr>
<tr>
<td>CE_7</td>
<td>215</td>
<td>15</td>
</tr>
<tr>
<td>CE_8</td>
<td>285</td>
<td>8</td>
</tr>
<tr>
<td>CE_9</td>
<td>380</td>
<td>13</td>
</tr>
<tr>
<td>CE_10</td>
<td>215</td>
<td>24</td>
</tr>
</tbody>
</table>
In simulations, the basic workload (jobs stream) is generated by using Lublin-Feitelson model [24], which is derived from the workload logs of real supercomputers. It consists of 10000 jobs, each is characterized by its arrival time, resource demands, and estimation of execution time. The resource demands of each job are enlarged by f times, where f is uniformly distributed in [10,20], so as to simulate the co-allocation in large-scale Grid environments. As the basic workload does not include deadline, we append each job with a deadline constraint as

\[ \text{deadline}_j = \text{arrival time}_j + k \cdot \text{execution time}_j \]

where k is a random variable that uniformly distributed in interval [1.5,5.5].

B. Performance Comparison

In the simulation, we compare the performance of four co-allocation policies in terms of resource utilization rate, violation rate, and resource benefits. The four policies are described briefly as following:

- **Round Robin Policy** [25] (RR_P): The meta-scheduler assigns jobs to clusters in turn. If the job's resource requirements cannot be meet for any single cluster, scheduler try to assigns the job to two or more clusters.
- **Capability-based Random Policy** [26] (CR_P): The probability of selecting a cluster for a job is proportional to its processors' speed and number. This policy is not just a purely random one, and it is driven by the intuition that more jobs should be assigned to more powerful clusters.
- **Cluster Minimized Policy** [10] (CM_P): The meta-scheduler tries to assign a job to a set of clusters, with arming at minimizing the size of the set.
- **Virtual Resource Agent Policy** (VRA_P): The meta-scheduler assigns a job to suitable VRAs firstly, and then VRAs allocate physical resources in its charge to execute the job. The algorithm is shown in algorithm 2. This is the co-allocation policy developed in this paper.

For the first three policies, there is no VRA entity in the system, so the resource prices in different clusters are fixed as listed in Table 1. For VRA_P, we set the initial \( p^* \) as the average value of all prices listed in Table II, and initial \( p_i \) for all VRAs is the same as \( p^* \). As described in section IV, a VRA adjust its price according to the difference between \( p_i \) and \( p^* \). We set that if \( p_i < p^* \), then \( V_i \) ’s price increases 10%, else decreases 10%. The adjustment of price is triggered each time when 200 jobs have been completed, so there are 50 chances for VRAs to adjust their resource prices. As to \( p^* \), the event of price adjusting is triggered each time when 500 jobs have been finished, so \( p^* \) will be adjusted for 20 times.

We set that if \( U_i^j < 0 \) and \( U_i^v < 0 \), the \( V_i \) would release 10 percent of its resources to a temp resource pool. Then, for those \( V_i \) satisfying \( U_i^j > 0 \) and \( U_i^v > 0 \), the resources in the temp resource pool will be fairly allocated to them.

As to VRA_P, its utilization is relative lower than CM_P for the first 2000 jobs. However, when the system is in a stable state, VRA_P’s utilization becomes higher than CM_P's, and does not fluctuate so dramatically as CM_P does. The reason is that: at first, VRA_P allocates resource based on jobs’ utility function, so a few powerful resources that can better meet jobs’ requirements will be frequently selected. That results in low resource utilization for those resources with low capacity. However, VRA_P is capable of adjusting its price scheme and VRAs’ size according to the benefits of both the system and clients. This feedback mechanism helps VRA_P find an efficient solution to organize the computing resource after a period of time. From Fig. 3,
we can also see that the average utilization of each CE is relatively balanced when using VRA_P.

In Table 2, we list the statistical information of the four policies after completing all the jobs in the workload. The details of each CE’s benefits are shown in Fig. 4. In simulation, we assume that whether accepting a job or not will depend on job’s budgets and resource prices. For the first three policies, as the resource prices are fixed, so the number of accepted jobs is the same. For VRA_P, the resource prices are adjusted dynamically at runtime, so the number of accepted jobs is different from the other three. Violation occurs if the system cannot actually meet a job’s deadline after completing the job. If deadline violation occurs, the system cannot get any benefits.

As we can see in Table 2, although the resource utilization of RR_P is the highest, its violation rate is still the highest too, which leads to its benefits in a very low level. CR_P takes into account the resource static capacity while allocating resource, which reduces the violation probability. However, CR_P suffers from load imbalance (shown in Fig. 4), so CM_P is more effective to meet jobs’ deadline than CR_P. As to VRA_P, it selects the resources that with an optimal probability to meet a job’s deadline requirement, so its violation rate is in a significant low level. In addition, VRA_P adjusts its price scheme according to the feedbacks from the system and the clients. So VRA_P can provide reliable QoS guarantees for applications in terms of budget and deadline, as well as an optimal price scheme for maximal system benefits.

C. QoS Performance with Different VRA_P Parameters

In order to further investigate the VRA_P’s QoS performance with different model parameter, we conduct a set of simulations with various combinations of the two key parameter $\Delta k$ and $\Delta p$. In VRA_P, $\Delta k$ and $\Delta p$ are the decrement or incremental of resource quantity and retail price, respectively. We conduct extensive simulations to examine the effects of both parameters on the performance of VRA_P in terms of resource benefits. The results are shown in Fig. 5.

In simulations, we test four different values of $\Delta p$ (5%, 10%, 15%, 20%) combining with gradually increasing $\Delta k$ from 2% to 20%. As shown in Fig. 5, the resource benefits are the most optimal when $16\%$ $\Delta k$ and $10\%$ $\Delta p$. When $\Delta p > 10\%$, the resource benefits becomes irregular with the increasing of $\Delta k$. It is because that large $\Delta p$ means the retail price fluctuate dramatically, as VRA_P tends to select those VRAs with low retail prices, so it is hard for the VRAs to obtain the optimal retail prices for maximizing the resource benefits. When $\Delta p = 5\%$, VRAs have to take a long time to get the optimal retail prices, so that the resource benefits can not be as much as the case when $\Delta p = 10\%$. According to the analysis in section IV.C, it can be explained as: it is difficult for the VRA-based model to entry into balancing state while using large value of $\Delta p$; On the other side, small value of $\Delta p$ makes the speed of convergence to balancing state too slow.

VI. CONCLUSION

In this work, we address this issue by presenting a novel two-phase co-allocation model. In the proposed model, we introduce the concept of Virtual Resource Agent, which is able to provide quantitative QoS guarantee for applications in terms of budget and deadline, as well as a feasible method to maximize the benefits of resource providers. We first analyze the model by using a three-participant game model and computing...
economy principle. Based on the theoretical analysis, a QoS-based co-allocation policy called VRA_P is proposed. Experiment results show that VRA_P can reduce deadline violation rate significantly, which in turn increases the benefits of the resource providers. This is useful for those grid applications with limited budget and stringent deadline in computing economy environment. In addition, we notice that VRA_P can implicitly achieve load balance by using price lever.

In this work, we mainly focus on computing resources co-allocation in multi-cluster Grid environment. We will take efforts to generalize our model and policy, and adapt to other resource types, for example bandwidth, storage and etc. Furthermore, we will consider combining our model with advance reservation to provide more reliable QoS guarantee for application’s deadline requirements. Also, we plan to define a SLA-based price bargain protocol in our framework.

ACKNOWLEDGMENT

This work was supported by a grant from the National Natural Science Foundation of China (No. 60970038). Authors gratefully acknowledge the Projects Supported by Scientific Research Fund of Hunan Provincial Education Department (08A009) for supporting this research. Project supported by Provincial Natural Science Foundation of Hunan (10JJ6099) supports the research. Project supported by Provincial Science & Technology plan project of Hunan (2010GK3048) supports the research.

REFERENCES


Peng Xiao was born in 1979. He received his master degree in Xiamen University in 2004. Now, he works in Hunan Institute of Engineering and is a Ph.D candidate in Central South University. His research interests include grid computing, parallel and distributed systems, network computing, distributed intelligence.

Dongbo Liu was born in 1974. He received his master degree in Hunan University in 2001. Now he works in Hunan Institute of Engineering and is a Ph.D candidate in Hunan University. His research interests include distributed intelligence, multi-agent systems, high-performance application.

Xilong Qu was born in 1978. He received his master degree in University of Electronic Science and Technology of China, and doctor degree in Southwest Jiaotong University. Currently, he is an associate professor in Hunan Institute of Engineering. His research interests include web service, distributed computing, information security technology.