Using Hybrid Gaming Model for Resource Co-allocation in Grid Environments

Xiao Peng* and Huang Zhe

Department of Computer and Communication, Hunan Institute of Engineering
**xtanefn@gmail.com; huangzhe197909@126.com*

Abstract

Recently, user-oriented QoS requirements have attracted more and more attentions, and the resource cost has become the key QoS attribute in many practical grid systems. However, how to decide the resource prices when co-allocating plenty of heterogeneous resources across different virtual organizations remains a challenging issue. In this work, we design and implement a novel co-allocation framework, in which the resource co-allocation procedure is divided into two phases. In the first phase, resource providers uses co-operative gaming model to decide their resource's original price, which can lead to maximal benefits for resource providers; In the second phase, non-cooperative gaming model is applied to find out the retail price (trading price) when users buy resources for executing their applications. Extensive simulations are conducted to evaluate the effectiveness and performance of the proposed framework, and the results show that the two-phase model can significantly improve the QoS satisfaction for those grid applications with constraint to limited budgets. Also, the efficiency of co-allocating multiple resources is also significantly improved comparing with existing policies.

Keywords: *Grid computing; Quality of service; Price mechanism; Gaming model*

1. Introduction

Grid systems and applications aim to integrating, virtualization, and managing resources and services within distributed, heterogeneous, dynamic virtual organizations [1]. Therefore, resource co-allocation becomes an important issue with increasing attention [2]. In grid environments, the co-allocation problem can be defined as the provision of allocation, configuration, and monitoring for the resource ensemble required by a single application. In conventional grid systems, co-allocation service mainly concentrates on system-oriented metrics, such as uniform programming interface, resource availability, load-balance and throughput. As more and more grid system began to provide service for public society, resource price become a critical issue when users want to execute their applications on given grid systems. Although many economic models have been proposed to solve the price decision problem, most of them can not take both resource provider and resource consumer into account at the same time, because their essential objectives are contradictory [3, 4].

In this work, we focus on the user-QoS-oriented co-allocation service, and our goal is to design an effective co-allocation framework, which can provide improved QoS

guarantee for those applications with constraint to limited budgets. In the proposed framework, we introduce a set of virtual resource agents for pricing mechanism and design a two-phase model for co-allocating plenty of resources from multiple virtual organizations. In our framework, virtual resource agents (VRA) is responsible for optimizing resources deployment and price scheme, and they also apply queuing system to model the working of resources for providing quantitative guarantee for application's deadline requirement.

The rest of this paper is organized as follows. Section 2 summarizes the existing work. In Section 3, the two-phase co-allocation framework is presented. In Section 4, we use gaming theory to analyze the proposed two-phase model. In Section 5, experiments are conducted to verify the effectiveness and performance of the proposed model. Finally, Section 6 concludes the paper with a brief discussion of our future planning.

2. Related Work

As the co-allocation service is always a fundamental infrastructure for distributed resource management as well as application scheduling, several co-allocation framework have been implemented for various real-work grid systems. For instance, the co-allocation framework in Legion system performed by a set of entities named as Enactor, which uses advance reservation service to ensure the resource availability when allocating underlying resources [5]. In the Globus systems [6], GARA framework is developed for providing atomic and interactive co-allocation strategies. In many multi-cluster systems (*i.e.*, DAS-2), KOALA scheduler has been widely used for co-allocating computing resources, which is characterized by its excellent fault tolerance functionality [7]. In [8], Waldrich, *et al.*, also developed a meta-scheduler, which relies on QoS negotiating mechanism to meet the user's various requirement including price, reliability, and execution time.

Besides the above co-allocation frameworks, many resource co-allocation models as well as policies have been proposed to optimize certain performance metrics, *i.e.*, response time, utilization, throughput, load balance. For instance, Leinberger et al. proposed two backfilling heuristics (FCFS/BB and FCFS/BL) for redundant resource co-allocation [9]. Their experiment results indicated that load balancing co-allocation policy outperforms classical policy such as FCFS/FF over 50% in terms of mean response time. In [10], Mohamed et al. proposed a Close-to-Files policy, which always try to schedule jobs on those computing sites that are most close to their input data, in this way, the communication overhead can be significantly reduced. To evaluate the performance of different co-allocation policies, Bucur and Epema [11-13] conducted massive real-word experiments in the grid testbed DAS-2 [14]. Based on their experimental results, they drew an important conclusion that workload-aware co-allocation policies are effective to reduce the mean response time as well as obtain better load-balance.

To the best of our knowledge, only a few of existing studies has addressed the issue of resource co-allocation for grid applications under the constraints of budget and deadline at the same time. For example, Nimrod-G is a famous grid system that uses computing economy driven architecture for managing resources and scheduling task [15]. In Nimrod-G, three

adaptive algorithms for deadline and budget constrained 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. Recently, game theory has been widely applied to solve the resource allocation problem in grid computing [17-19]. In those studies, it is often assumed that participants in games are selfish, and the methods are to find the equilibrium solution of resource price or allocation scheme. To comparing those game theory based methods, Khan, *et al.*, classifies them as cooperative, semi-cooperative and non-cooperative in [19]. By extensive simulations, Khan indicated that agent-based cooperation gaming model seems performing better than others. However, Khan's cooperative model is of very high computational complexity, which inspires us to find more efficient method.

3. Co-allocation Framework with Virtual Resource Agents

The co-allocation framework proposed in this work is demonstrated in Figure 1, in which there are a set of computing elements (CEs) each representing a high-performance computing cluster, a set of Virtual Resource Agents (VRA), and a meta-scheduler for global task dispatching.

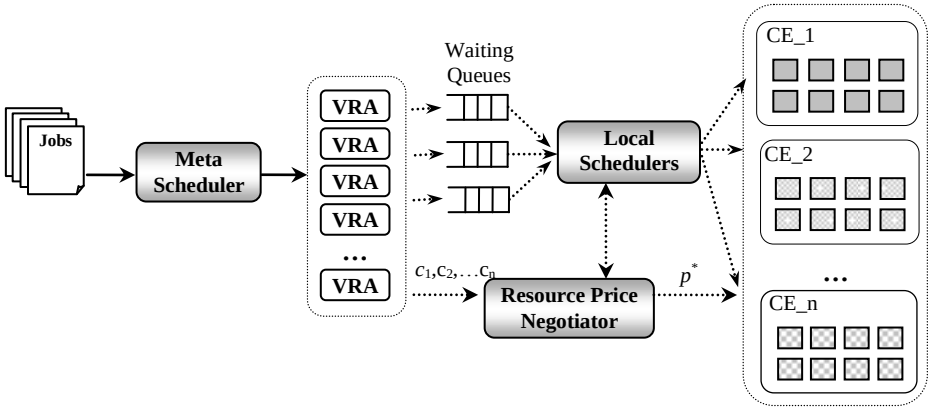

Figure 1. VRA based Co-allocation framework

As shown in Figure 1, the meta-scheduler is responsible for selecting suitable VRAs according to user's QoS constraints (*i.e.*, budget and deadline) and dispatching them to those selected VRAs. Unlike existing co-allocation framework, we introduce a set of VRA entities in this framework, which are designed to obtain resource from different CEs on a uniform price p^*(details on "uniform price" will be described in Section 4) through the *resource price negotiator* component and sell them to users at different retail prices. As the VRAs are virtual entities, they can dynamically change their resource quantity (noted as $<c_1,c_2,\ldots,c_n>$) at runtime when obtaining resources form CEs. In this way, VRAs provides a reasonable resource pricing mechanism to satisfy user's budget limitation and maximize provider's benefits as much as possible.

It is clear that there are three classes of participants in the co-allocation framework: the resource providers (CEs), users, and VRAs. To analyze the optimal pricing scheme, we set that the VRAs and CEs have the same objective, that is, maximizing resource utilization and benefits. So, their relationship can be modeled as cooperative gaming model. On the other side, the relationship between users and VRAs is non-cooperative in nature because users always want to minimize their resource costs. According the above assumption, it can be seen that the three classes of participants is organized as a three-side co-allocation model which is similar to the classical *Producer-Retailer-Client* model [21]. In the following sections, our main objective is to proof the validity and figure out the solution of this two-phase gaming model.

4. Solutions of the Three-side Gaming Model

4.1 Utility Functions

To analyze the two-phase three-side co-allocation model, we first need to formulate the utility functions of the three classes of participants, respectively. As mentioned in the last section, VRAs need to obtain a certain number of resources from CEs at a uniform price p^*, and all the resources will be allocated to users through VRAs. Therefore, the utility function of each CE can be defined as its total benefits that shown as following

$$U^S = p^* \cdot \sum_{i=1}^{n} s_i \tag{1}$$

where s_i is the total quantity of available resources in CE_i. In this work, we ignore the price difference of various resources at the first phase because the retail price will be decided when users buy resources at the second phase. Therefore, the actual trading price will be determined by VRAs and users in the non-cooperative gaming model. From the perspective of the resource providers, they only needs to care about the total benefits that can be easily tuned by adjusting p^*. This idea is inspired by *Producer-Retailer-Client* model.

From the perspective of VRA, their costs can be noted as $p^* c_i$, and their gross benefits can be noted as $c_i p_i \rho_i$, where c_i be the quantity of resources in the virtual resource agent V_i, ρ_i be the real utilization in V_i, p_i is the retail price that decided by individual VRA. Therefore, the utility function of V_i can be defined as

$$U_i^V = c_i \cdot p_i \cdot \rho_i - p^* \cdot c_i \tag{2}$$

From the perspective of grid system, we also need to calculate the whole benefits of all the VRAs, which are defined as following

$$U^V = \sum_{i=1}^{n} (c_i \cdot p_i \cdot \rho_i) - p^* \cdot \sum_{i=1}^{n} c_i \tag{3}$$

In this work, we mainly concentrate on two most mentioned QoS attributes: costs and deadline. So, the user's job j can be characterized as a 3-tuple: $< r_j, b_j, d_j >$, where r_j is

the resource requirements, b_j is the budget limitation, d_j is the absolute deadline. Let J be the set of VRAs being allocated to job j, and r_j^i be the amount of resources allocated from V_i to execute job j, then the cost of job j is $\sum_{i\in J}(r_j^i \cdot p_i)$. As the deadline guarantee is not a quantitative measurement, we model it as a probability. Let $\mathbf{E}_i$ 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 $\prod_{i\in J}\Pr\{\mathbf{E}_i\}$. So we define the utility function of job j as follows

$$U_j^C = \prod_{i\in J}\Pr\{\mathbf{E}_i\} / \sum_{i\in J}(r_j^i \cdot p_i) \tag{4}$$

4.2. Analysis of Cooperative Gaming Model

As mentioned above, Both VRAs and CEs represent the objective of resource providers, that is maximizing resources utilization and resource benefits, so we use cooperative model to describe their relationship. In the cooperative model, a solution can be described as a pair $< p^*, \mathbf{C} >$, where $\mathbf{C} = (c_1, c_2 \ldots c_n)$ is a vector representing the resource quantity in each VRA.

Considering the current price set by the CEs is p^*, by utilities defined in (2) and (3) we have the VRAs' benefits noted as $(U_1^V, U_2^V, \cdots, U_n^V)$, and the total VRAs benefits U^V. If $U^V > 0$, that means 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. Therefore, the CEs can adjust the uniform price as $p_1^* = (U^S + U^V) / \sum_{i=1}^{n} s_i$. 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 $(U_1^{V'}, U_2^{V'}, \cdots, U_n^{V'})$. From the definition of U_i^V, we can known that if $U^V > 0$ then $\forall i \in [1 \ldots n]$ $U_i^V > U_i^{V'}$, which means increasing p^* to p_1^* will decrease the benefits for all VRAs. Thus, there are three cases we should consider:

- $U_i^V > 0$ and $U_i^{V'} > 0$: In this case, the benefits of V_i is positive even it had to pay a higher price for resources, which means more resources should be allocated to V_i.
- $U_i^V > 0$ and $U_i^{V'} < 0$: 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.
- $U_i^V < 0$ and $U_i^{V'} < 0$: 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 $(p_1^*, \mathbf{C}')$, where p_1^* is the new resource price decided by the system, $\mathbf{C} = (c_1', c_2' \ldots c_n')$ 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 $< p^*, (c_1, \ldots c_n) >$, in which $S^+(p^*) = \{V_i \mid U_i^V > 0\}$ is the set of VRAs with positive benefits at p^*;

$S^{-}(p^{*})=\{V_i \mid U_i^V<0\}$ is the set of VRAs with negative benefits; $S^{0}(p^{*})=\{V_i \mid U_i^V=0\}$ is the set with zero benefits.

Algorithm 1: Obtain $<p^{*},(c_1,\ldots c_n)>$ as cooperative gaming model solution
Begin
1. $p_1^{*}=\frac{1}{n}\sum_{i=1}^{n}(p_i\cdot\rho_i)$
2. **for** $i=1$ to n
3. calculate $U_i^{V'}$ at the new price p_1^{*}
4. **if** $V_i\in S^{+}(p_1^{*})$ **then**
5. add i to *expand_list*
6. **else if** $V_i\in S^{-}(p_1^{*})$
7. add i to *shrink_list*
8. **end if**
9. **end for**
10. **for each** i in *shrink_list*
11. $c_i^{'}=(1-\Delta k)\cdot c_i$
12. find j in *expand_list*, which satisfying one of the two conditions: (1) the amount of resources come from CE_i in V_i is maximal; (2) U_j^V is maximal in the VRAs that need to be expanded.
13. $c_j^{'}=c_j+\Delta k\cdot c_i$
14. **end for**
End

4.3. Analysis of Non-Cooperative Gaming Model

The users always tend to select the VRAs with lower retail price. Although high retail price can bring better benefits for VRAs, their resource utilization will be reduced too. On the other side, 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 $<p^{*},(c_1,\ldots c_n)>$. So, the solution of no-cooperative model is the retail prices $(p_1,\ldots,p_n)$. 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 $p_i(\rho_i)$. Therefore, the utility function of V_i can be formulated as

$$U_i^V=c_i\cdot p_i(\rho_i)\cdot\rho_i-p^{*}\cdot c_i \tag{5}$$

Let $dU_i^V/d\rho_i=0$, we can get the equation (6). Denote the solution of equation (6) as ρ_i^{*}. It is clear that the maximal value of U_i^V can be obtained when $\rho_i=\rho_i^{*}$. So we call ρ_i^{*} as the optimal resource utilization rate of V_i.

$$p_i^{'}(\rho_i)\cdot\rho_i+p_i(\rho_i)=0 \tag{6}$$

In the practical grid system, V_i cannot set its ρ_i as ρ_i^{*} by itself because it is measured instead of being adjusted. However, V_i can change its retail price to effect the user's resource selection. Through comparing the difference between ρ_i^{*} and ρ_i, a VRA can

decide whether increasing or decreasing its retail price according their objectives. For example, if $\rho_i < \rho_i^*$ then V_i can decrease its price, else the V_i would increase its price. This process can be performed repeatedly until an optimal retail price scheme is obtained.

Finally, we analyze the probability that the deadline constraint of a job can be guaranteed, which is shown in (4) and noted as $\prod_{i\in J}\Pr\{\mathbf{E}_i\}$. In this work, it is assumed that the arrival of jobs in V_i follows Poisson distribution with rate λ_i, and the execution time follows Exponential distribution with rate μ_i. Therefore, a VRA can be modeled as a $M/M/c_i$ queuing system [22]. So, the utilization rate of V_i can be expressed as $\rho_i = \lambda_i/(c_i \cdot \mu_i)$. In this paper, we only consider the case $\rho_i < 1$. Let $\boldsymbol{\psi}_i$ be a random variable representing the number of waiting jobs in V_i. According to queuing theory, the probability that there are k waiting jobs in V_i is

$$\Pr\{\boldsymbol{\psi}_i = k\} = \begin{cases} \delta \cdot \dfrac{\rho_i^{k+c_i} \cdot c_i^{c_i}}{c_i!}, & k > 0 \\ \displaystyle\sum_{n=0}^{c_i} \delta \cdot \frac{(\rho_i \cdot c_i)^n}{n!}, & k = 0 \end{cases} \tag{7}$$

$$\delta = \left[\sum_{n=1}^{c_i} \frac{(\rho_i \cdot c_i)^n}{n!} + \frac{(\rho_i \cdot c_i)^{c_i}}{c_i} \frac{1}{1-\rho_i} \right]^{-1} \tag{8}$$

Let $\boldsymbol{\omega}_i$ be a random variable representing the completion time (waiting and execution time) of a job, then the probability that V_i can provide deadline guarantee for the job is noted as

$$\Pr\{\mathbf{E}_i\} = \Pr\{\boldsymbol{\omega}_i \le d_j\} \tag{9}$$

According to the theory of $M/M/c_i$ queuing model, the service rate is $c_i\mu_i$, which means the number of jobs that the resource can finished is $c_i\mu_i$ jobs in a time unit. So, the number of jobs that V_i can complete in period d_j is $c_i\mu_i d_j$. Therefore, the probability that V_i can guarantee a job's deadline d_j is equal to the probability that the waiting jobs in V_i is not more than $c_i \cdot \mu_i \cdot d_j - 1$. By (7)(8)(9), we can get that

$$\begin{aligned} \Pr\{\mathbf{E}_i\} = \Pr\{\boldsymbol{\omega}_i \le d_j\} = \Pr\{\boldsymbol{\psi}_i \le c_i \cdot \mu_i \cdot d_j - 1\} &= \sum_{k=0}^{c_i \cdot \mu_i \cdot d_j - 1} \Pr\{\boldsymbol{\psi}_i = k\} \\ = \sum_{k=1}^{c_i \cdot \mu_i \cdot d_j - 1} \delta \cdot \frac{\rho_i^{k+c_i} \cdot c_i^{c_i}}{c_i!} &+ \sum_{n=0}^{c_i} \delta \cdot \frac{(\rho_i \cdot c_i)^n}{n!} \end{aligned} \tag{10}$$

In this work, we use this deadline guarantee to define the user's utility function so as to reflect the QoS requirement of real-time applications. Therefore, the equation (4) can be formulated as following

$$U_j^C = \frac{1}{\sum_{i\in J}(r_j^i \cdot p_i)} \prod_{i\in J} \left(\sum_{k=1}^{c_i \cdot \mu_i \cdot d_j - 1} \delta \cdot \frac{\rho_i^{k+c_i} \cdot c_i^{c_i}}{c_i!} + \sum_{n=0}^{c_i} \delta \cdot \frac{(\rho_i \cdot c_i)^n}{n!} \right) \tag{11}$$

The meta-scheduler selects the best VRAs to execute jobs according to formula (11). In the practice grid systems, 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.

5. Experiments and Performance Analysis

5.1. Experiment configurations

In the experiments, the GridSim [23] is used to construct the simulative platform to evaluate the performance of the proposed framework. The simulative platform is a multi-cluster computational grid, in which there consists of ten high-performance computing clusters. In order to make the simulative platform more close to practical grid system, we configure the platform's parameters based on the large-scale grid testbed DAS-2 [14], and the detailed setting of each cluster is listed in Table 1.

Table 1. Grid Settings in Simulation

CEs	Processor number	MIPS / processor	Price / MIPS
CE_1	70	377	12
CE_2	120	410	22
CE_3	200	380	35
CE_4	150	285	17
CE_5	110	285	17
CE_6	50	515	25
CE_7	120	215	15
CE_8	80	285	8
CE_9	300	380	13
CE_10	160	215	24

To generate the testing workload, we use the widely-applied Lublin-Feitelson model [24], which is derived from the long-term trace logs in real-world supercomputer centers. The basic workload has 10,000 independent jobs, each having three attributes: resource requirements, arrival time, and execution time. As the resource requirements of basic workload are too low, we amplify this attribute f times for each job with aiming to simulate real-world grid applications. In addition, as the basic workload does not include the deadline constraint, we append each job with a deadline attribute according to the following formula

$$deadline_j = arriveal\_time_j + k \cdot execution\_time_j \tag{12}$$

where k is a random variable that uniformly distributed in interval [1.5, 5,5].

5.2. Performance Evaluation and Comparison

To investigate the validity and effectiveness of the proposed co-allocation framework and the corresponding algorithm, we compare our co-allocation policy (Virtual Resource Agent Policy, VAR_P) with other three policies, including *Round Robin Policy* (RR_P) [25], *Capability-based Random Policy* (CR_P) [26], and *Cluster Minimized Policy* (CM_P) [10]. The performance metrics include resource benefits, resource utilization rate, and violation rate. When using our VRA_P, we set the original p^* as the mean value of all prices listed in Table 1, and initial p_i of VRAs is the same

as p^*. As mentioned in Section 4, the VRA can adjust the retail price according to the difference between ρ_i and ρ_i^*. In the experiment, we set that if $\rho_i < \rho_i^*$ then V_i 's price will be increased by 10 percent, else it will be decrease by 10 percent. The price adjustment is triggered when each 200 jobs have been completed, so there are 50 chances to adjust their resource prices for each VRA. As to p^*, the event of price adjusting is triggered each time when 500 jobs have be finished, so p^* will be adjusted for 20 times. We set that if $U_i^V < 0$ and $U_i^{V'} < 0$, V_i will release 10 percent of its resources to CEs. Then, for those V_i satisfying $U_i^V > 0$ and $U_i^{V'} > 0$, the resources of CEs will be allocated to them. In the Figure 2, we illustrate the real-time resource utilization of the four co-allocation policies; in the Figure 3, we illustrate the average resource utilization rate of each CE.

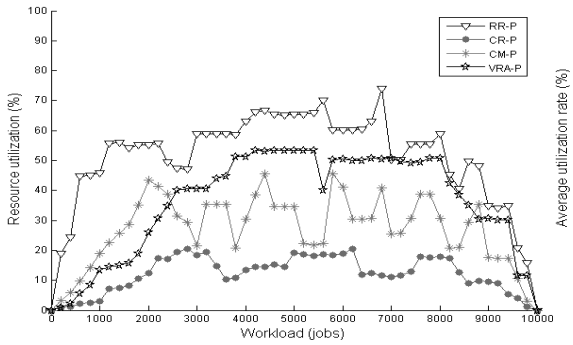

Figure 2. Real-time utilization

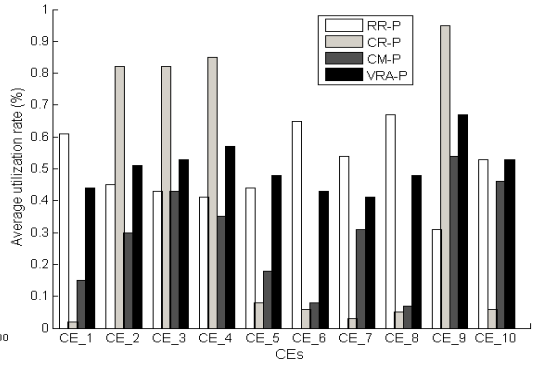

Figure 3. Average resource utilization

As we can see from the results, the utilization of RR_P is the highest with average value 53%, the lowest is CR_P with average value 12%. The reason is that RR_P co-allocates resources to jobs in turn, which is able to quickly full-utilize resources in a balancing fashion as shown in Figure 3. For CR_P, it co-allocates resources to tasks based on the static capability of underlying resources. Therefore, most workload is assigned to those powerful resource sites (*i.e.*, CE_2, CE_3, CE_4, CE_9). This policy make the CR_P tends to be load-imbalance relative to other policies.

As to our VRA_P, the resource utilization is slightly lower than CM_P for the first 2,000 jobs. However, when the simulations is in a stable state, resource utilization of VRA_P is significantly higher than CM_P, and does not fluctuate so dramatically as CM_P does, which can be seen in the Figure 2. Such a result can be explained as: firstly, VRA_P co-allocates resources based on user's utility function; therefore, a few powerful resource sites that can better meet user's requirements will be selected more likely. That results in low resource utilization for those resources with low capability. However, VRA_P is capable of adjusting its price scheme and quantity of owned resources according to the benefits of both the grid system and user's requirements. Such a feedback mechanism makes the VRA_P quickly find an efficient solution to organize the computing resources when the system works in stable phase. From Figure

3, we can easily see that the utilizations of CEs are relative balanced when using VRA_P to co-allocate resources.

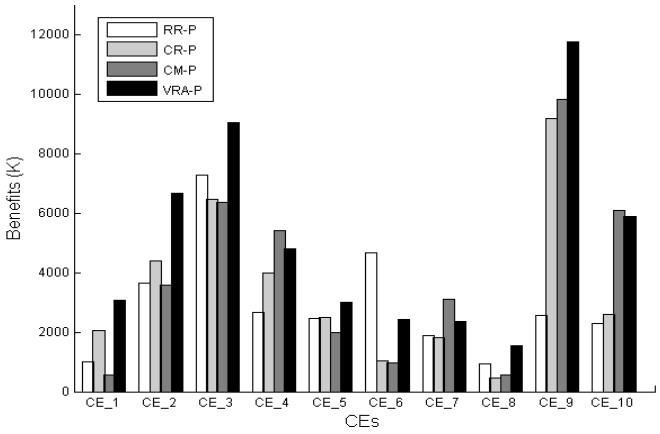

Figure 4. Total resource benefits of each CE with different policies

The details of each CE's benefits are shown in Figure 4, and the deadline violation rate, accepted job numbers, and total resource benefits are shown in Table 2. In the experiments, we set that whether accepting a job or not depend on each user's budget limitation and resource prices. For VRA_P, the resource prices are adjusted dynamically at runtime, so the number of accepted jobs is different from the other policies. As the resource prices are fixed when using other three policies, the number of accepted jobs is the equal. Deadline violation occurs if the grid platform can not meet a job's deadline after completing the job. We also set that if the deadline violation occurs, the grid system can not charge any costs from the user.

Table 2. Violation rate and resource benefits

Policy	Accepted jobs	Violation rate	Resource benefits(K)
RR_P	9648	26%	29413
CR_P	9648	13%	34503
CM_P	9648	11%	38487
VRA_P	9375	3.5%	50545

As shown in Table 2, the resource utilization of RR_P is the highest among all policies, however, its violation rate is also the highest, which makes its resource benefits very low. It is because that we set if the user's requirement can not be meet, grid system can not charge with users. CR_P takes into account the resource static capability while co-allocating resources, which reduces the probability of deadline violation. However, CR_P suffers from load-imbalance, so CM_P is more effective to meet user's deadline than CR_P. As to VRA_P, it tends to select those resource sites that have an optimal probability to meet user'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.

5.3. QoS Performance with Different VRA_P Parameters

In VRA_P, Δk and Δp are the decrement/incremental of resource quantity and retail price. In order to further investigate the VRA_P's QoS performance with different model parameter, we conductive a set of simulations with various combinations of the two key parameter Δk and Δp. The results are shown in Figure 5 and Figure 6.

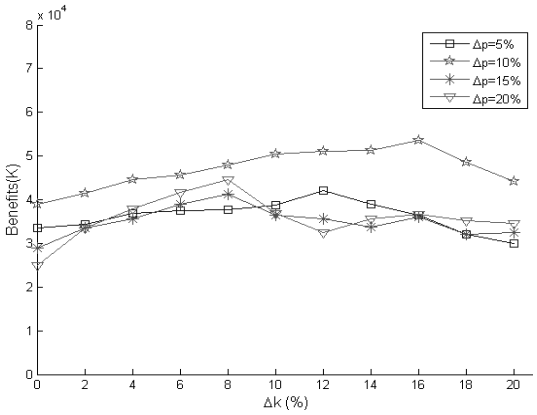

Figure 5. Effects of Δk & Δp on resource benefits

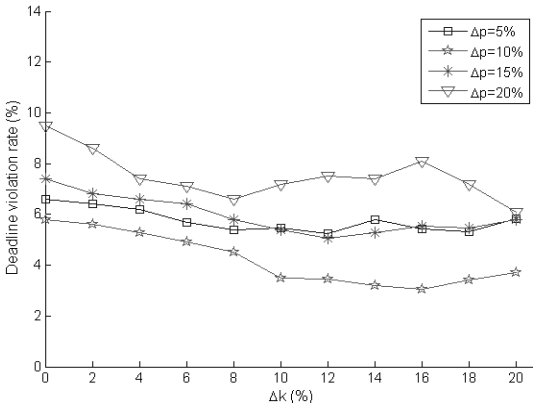

Figure 6. Effects of Δk & Δp on violation rate

In simulations, we test four different values of Δp (5%, 10%, 15%, 20%) combining with gradually increasing Δk from 2% to 20%. As shown in Figure 5 and Figure 6, the violation rate and resource benefits are the most optimal when $\Delta k = 16\%$ and $\Delta p = 10\%$. When $\Delta p > 10\%$, the resource benefits becomes irregular with the increasing of Δk. It is because that large Δp 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 4, it can be explained as: it is difficult for the VRA-based model to entry into balancing state while

using large value of Δp ; On the other side, small value of Δp makes the speed of convergence to balancing state too slow.

6. Conclusion

In this work, we address this issue by presenting a novel VRA-based 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 improve the resource benefits. We first analyze the model by using a three-side model. Based on the theoretical analysis, a co-allocation policy called VRA_P is proposed. Experimental results show that VRA_P can reduce deadline violation rate significantly, which in turn increases the resource benefits. In the future, we will take efforts to generalize the proposed model and policy to other resource types, i.e. bandwidth, storage and etc. Furthermore, we also plan to combine our model with advance reservation to provide more reliable QoS guarantee for applications with soft real-time deadline.

Acknowledgements

This work is supported by the Provincial Science & Technology plan project of Hunan (No.2012GK3075).

References

[1] I. Foster and C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure", Second Edition. Elsevier Inc., San Francisco, **(2004)**.
[2] M. Netto, C. Vecchiola, M. Kirley and Rajkurnar, "Use of Run Time Predictions for Automatic Co-allocation of Multi-cluster Resources for Iterative Parallel Applications", Journal of Parallel and Distributed Computing, vol. 10, no. 71, **(2011)**, pp. 1388-1399.
[3] C. Courcoubetis and R. Weber, "Economic Issues in Shared Infrastructures", IEEE/ACM Transactions on Networking, vol. 2, no. 20, **(2012)**, pp. 594-608.
[4] A. Haque, S. M. Alhashmi and R. Parthiban, "A Survey of Economic models in grid computing", Future Generation Computer Systems, vol. 8, no. 27, **(2011)**, pp. 1056-1069.
[5] A. Natrajan, M. Humphrey and A. Grimshaw, "The Legion Support for Advanced Parameter-space Studies on a Grid", Future Generation Computer Systems, vol. 8, no. 18, **(2002)**, pp. 1033-1052.
[6] I. Foster, "Globus Toolkit version 4: Software for Service-oriented Systems", Journal of Computer Science and Technology, vol. 4, no. 21, **(2006)**, pp. 513-520.
[7] H. Mohamed and D. Epema, "KOALA: a Co-allocating Grid Scheduler", Concurrency and Computation-Practice & Experience, vol. 16, no. 20, **(2008)**, pp. 1851-1876.
[8] O. Waldrich, P. Wieder and W. Ziegler, "A Meta-Scheduling Service for Co-allocating Arbitrary Types of Resources", CoreGRID Technical Report TR-0010, **(2005)**.
[9] W. Leinberger, G. Karypis and V. Kumar, "Job Scheduling in the presence of Multiple Resource Requirements", ACM/IEEE International Conference on Supercomputing, Cancun, Mexico, **(1999)** May 1-5, pp. 1-9.
[10] H. H. Mohamed and D. H. J. Epema, "An Evaluation of the Close-to-Files Processor and Data Co-Allocation Policy in Multiclusters", ACM/IEEE International Conference on Cluster Computing, San Diego, CA, USA **(2004)** September 20-23, pp. 287-298.
[11] A. I. D. Bucur and D. H. J. Epema, "Scheduling Policies for Processor Coallocation in Multicluster System", IEEE Transaction on Parallel and Distributed Systems, vol. 7, no. 18, **(2007)**, pp. 958-962.
[12] A. Iosup and D. Epema, "Grid Computing Workloads", IEEE Internet Computing, vol. 2, no. 15, **(2011)**, pp. 19-26.
[13] O. O. Sonmez, H. Mohamed and D. H. J. Epema, "On the Benefit of Processor Coallocation in Multicluster Grid Systems", IEEE Transactions on Parallel and Distributed Systems, vol. 6, no. 21, **(2010)**, pp. 778-789.
[14] J. Andreeva, M. Boehm and B. Gaidioz, "Experiment Dashboard for Monitoring Computing Activities of the LHC Virtual Organizations", Journal of Grid Computing, vol. 2, no. 8, **(2010)**, pp. 323-339.

[15] D. Abramson, R. Buyya and J. Giddy, "A Computational Economy for Grid Computing and its Implementation in the Nimrod-G Resource Broker", Future Generation Computer Systems, vol. 8, no. 18, **(2002**), pp. 1061-1074.
[16] L. Rodero-Merino, E. Caron, A. Muresan and F. Desprez, "Using Clouds to Scale Grid Resources: An Economic Model", Future Generation Computer Systems, vol. 4, no. 28, **(2012)**, pp. 633-646.
[17] Y. -K. Kwok, K. Hwang and S. Song, "Selfish Grids: Game-Theoretic Modeling and NAS/PSA Benchmark Evaluation", IEEE Transaction on Parallel and Distributed Systems, vol. 5, no. 18, **(2007)**, pp. 621-636.
[18] D. Niyato, A. V. Vasilakos and Z Kun, "Resource and Revenue Sharing with Coalition Formation of Cloud Providers: Game Theoretic Approach", IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, May 23-26 2011, Newport Beach, CA, USA, **(2011)**, pp. 215-224.
[19] S. U. Khan and I. Ahmad, "Non-cooperative, Semi-cooperative, and Cooperative Games-based Grid Resource Allocation", IEEE International Symposium on Parallel and Distributed Processing, Rhodes Island, Greece, **(2006)** April 25-29, pp. 1-9.
[20] A. J. Zaliwski, "In Search of Visualization Metaphors for PlanetLab", IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, May 17-20, 2010, Melbourne, Australia, **(2010)**, pp. 583-584.
[21] P. Ghosh, K. Basu and S. K. Das, "A Game Theory-based Pricing Strategy to Support Single/Multiclass Job Allocation Schemes for Bandwidth-constrained Distributed Computing Systems", IEEE Transactions on Parallel and Distributed Systems, vol. 3, no. 18, **(2007)**, pp. 289-306.
[22] D. Gross and C. M. Harris, "Fundamentals of Queuing Theory 3rd Edition", John Wiley and Sons, **(1998)**.
[23] A. Sulistio, U. Cibej, S. Venugopal, B. Robic and R. Buyya, "A Toolkit for Modelling and Simulating Data Grids: an Extension to GridSim", Concurrency and Computation-Practice & Experience, vol. 13, no. 20, **(2008)**, pp. 1591-1609.
[24] U. Lublin and D. G. Feitelson, "The Workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Jobs", Journal of Parallel and Distributed Computing, vol. 11, no. 63, **(2003)**, pp. 1105-1122.
[25] C. L. Dumitrescu, I. Raicu and I. Foster, "The Design, Usage, and Performance of GRUBER: A Grid Usage Service Level Agreement based Brokering Infrastructure", Journal of Grid Computing, vol. 1, no. 5, **(2007)**, pp. 99-126.
[26] V. Berten, J. Goossens and E. Jeannot, "On the Distribution of Sequential Jobs in Random Brokering for Heterogeneous Computational Grids", IEEE Transaction on Parallel and Distributed Systems, vol. 2, no. 17, **(2006)**, pp. 113-124.

Authors

Xiao Peng received the Ph.D degree in computer science in CSU at 2010. He is Currently an associate professor in the Hunan Institute of Engineering. His research interests include grid computing, distributed resource management. He is a member of ACM and IEEE.

Huang Zhe received his master degree in Xiangtan University in 2006. Now he is working as lecturer in Hunan Institute of Engineering, and as a network engineer in HP high-performance Lab. His research interests include distributed computing, system reliability evaluation, cloud computing. He is now a member of CCF in China.