International Journal of Future Generation Communication and Networking
Vol.7, No.3 (2014), pp.49-60
http://dx.doi.org/10.14257/ijfgcn.2014.7.3.05

Co-scheduling Deadline-Sensitive Applications in Large-scale Grid Systems

Dongbo Liu and Ning Han

School of Computer and Communication, Hunan Institute of Engineering
liudongbo74@126.com

Abstract

In large-scale grid systems, plenty of applications are constrained by soft or hard deadline requirement. However, it is difficult to guarantee the deadline requirements of these applications because of the dynamical nature of distributed systems. In this paper, a novel approach is proposed to evaluate the deadline-guarantee of co-allocation schemes that obtained from conventional co-allocation policies. Based on this approach, a hybrid-policy co-allocation model is also proposed to address the issue of deadline-constrained resource co-allocation in grid environments. The proposed model integrates multiple co-allocation policies to generate different co-allocation schemes, and selects the one with optimal deadline-guarantee for grid applications. By this way, the hybrid-policy model combines the merits of different policies, and overcomes the shortcomings of those policies. Extensive simulations are conducted to verify the effectiveness and the performance of the proposed model in terms of deadline-miss rate. Experimental results show that it can provide co-allocation scheme with enhanced deadline-guarantee as well as lower deadline-miss rate.

Keywords: *Grid Computing; Resource Co-Allocation; Quality of Service; Distributed Scheduling*

1. Introduction

Grid computing [1] has emerged as promising distributed platforms that aggregate heterogeneous resources for high-end applications, which frequently require multiple resources from different sites. Therefore, resource co-allocation is always the key issue in grid systems [2, 3]. At present, studies on co-allocation mainly focus on *co-allocation framework* [2-6] and *co-allocation policy* [7-12]. With the development of grid middleware, many effective co-allocation frameworks have be developed and deployed in practical grid systems. However, studies on co-allocation policy are relative left behind, especially in the area of QoS-based co-allocation policy [2, 13].

In the QoS requirements, *deadline-guarantee* is one of the requirements that frequently be mentioned [9, 13]. The difficulties of providing deadline-guaranteed when co-allocating resources are as following: (1) Lacking of a general measurement to evaluate the capability of heterogeneous resources, which makes it difficult to find the optimal co-allocation scheme for guaranteeing user's deadline requirement [2, 14]. (2) Workload on resources is unpredictable, which makes the capability-based co-allocation policies perform poor in practice [2, 9, 15-18]. In this paper, a novel approach is proposed to evaluate the deadline-guarantee of a specific co-allocation scheme that obtained from existing co-allocation policies. Based on this approach, a *Hybrid-Policy Co-allocation Model* (HPCM) is developed, which takes the advantages of existing co-allocation policies to generate multiple co-allocation schemes and select the scheme that can provide optimal deadline-guarantee for grid applications.

The rest of this paper is organized as follows: Section 2 presents the related work; Section 3 introduces the approach to evaluate the deadline-guarantee degree of a specific co-allocation scheme. Section 4 presents the design of HPCM prototype. In Section 5, simulations are conducted to verify the effectiveness and performance of the proposed model. Finally, Section 6 concludes the paper with a brief discussion of future work.

2. Related Work

As co-allocation has been a fundamental infrastructure for resource management and task scheduling in grid environments, many co-allocation frameworks have been developed. In Legion [4], co-allocation is supported by a set of entities called as *Enactor*. In EveryWare [5], *Lingua-Franca* is set of resource control interfaces for co-allocating heterogeneous resources across administrative domains. In Globus Toolkit [6], co-allocation is supported by *GARA* [2, 3], which incorporates both "atomic" and "interactive" co-allocation strategies. All these frameworks rely on *advance reservation* [3] for providing co-allocation services. However, advance reservation can only ensure the availability of resources at required time, but cannot provide guarantees to deadline [3, 15, 19-22]. In addition, many studies [16, 17] show that excessive advance reservation will degrade the system's performance, for instance, high rejection rate and low resource utilization.

To improve the performance of co-allocation frameworks, many co-allocation policies and models are proposed. In [7], the authors propose two backfilling-based heuristics (*FCFS/BB* and *FCFS/BL*) for K-resource co-allocation. Their simulations show that load balancing policy outperforms the others (i.e. *FCFS/FF*) over 50% in terms of mean response time. In [8], Mohamed *et al.* propose a *Close-to-Files* co-allocation policy, which tries to place tasks on resources that are close to the input files with aiming to reducing communication overhead. In [9], He-Ligang *et al.* address co-allocation for *non-real-time* and *soft real-time* tasks in multicluster grid. Two policies (*ORT* and *OMT*) are developed by numerically solving two optimization equation sets. The ORT aims to optimize *mean response time* for non-real-time tasks, and the OMR is to achieve the optimized *mean deadline-miss rate* for soft real-time tasks. To evaluate the performance of various co-allocation policies, the researchers conduct extensive experiments in large-scale grid testbed DAS-2 [10-12, 23]. Based on their experimental results, they draw an important conclusion that *workload-aware* co-allocation policies are more effective to reduce the mean response time and obtain better load-balance.

In all above studies, only the OMR policy proposed by He-Ligang *et al.* takes deadline-guarantee into account. However, the OMR policy is designed to obtain the *optimal average deadline-miss rate*, which means that it can not provide deadline-guarantee for an individual task. In addition, the OMR needs to solve a set of complex equations to obtain the optimal solution, which seems difficult to be applied in practical grid systems. In this paper, we take the dynamic workload on resources into account and use queueing system [20] to describe the working of grid resources. In this way, we avoid to solving the optimization equation sets as in the OMR.

In many studies, researches have applied queueing system to describe the working of grid resources. For example, Sun Xian-He et al. used queueing theory to predicate the availability of grid resources [17]; Wu Ming et al. applied M/G/1 queueing system to analyze the effects of advance reservation on local task's scheduling [19]; Berten et al. uses M/M/C queueing system to study the capability-based allocation policy in multicluster grid [14]. These studies show that queueing system is capable of precisely describing the working and workload model of grid resources in dynamic environments.

3. Real-time Co-Scheduling Framework

In grid environments, applications can be categorized into two types: *Parameter Sweep Application* (PSA), *Directed Acyclic Graph Application* (DAGA). In PSA, the sub-tasks are independent and do not communicate with each other. As to DAGA, it is usually represented by a directed acyclic graph, in which the nodes represent sub-tasks and the directed edges represent the executing order of the sub-tasks. In addition, *Redundant Request* (RR) [25] is a common technique that frequently being used in grid environments. By using RR technique, grid systems send multiple copies of a task onto different resources with aiming to improve the reliability of task scheduling or reduce task's execution time. So, Combining the task types and whether using RR technique, there are four scenarios should be taken into consideration when co-allocating resources, that as following: (1) *PSA_NRR*: Tasks are PSA, and the system does not use RR. (2) *PSA_RR*: Tasks are PSA, and the system uses Redundant RR. (3)

DAGA_NRR: Tasks are DAGA, and the system does not use RR. (4) *DAGA_RR*: Tasks are DAGA, and the system uses RR. In this paper, we only consider the PSA tasks, because the DAGA tasks are often mapped into workflows that are often scheduled by certain workflow-engine, which are out of the scope of this paper. So, the following sections will concentrate on co-allocation in the first two scenarios (PSA_NRR and PSA_RR).

3.1. Co-scheduling Model

As shown in Figure 1, a typical computing grid consists of a *meta-scheduler*, *local schedulers*, and *computing sites* (*i.e.*, cluster or MPP). On receiving new tasks, the meta-scheduler is responsible for co-allocating resources for them by using certain co-allocation policy. The local schedulers allocate underlying resources for the sub-tasks according to co-allocation schemes. Therefore, meta-scheduler is the key component when co-allocating resources.

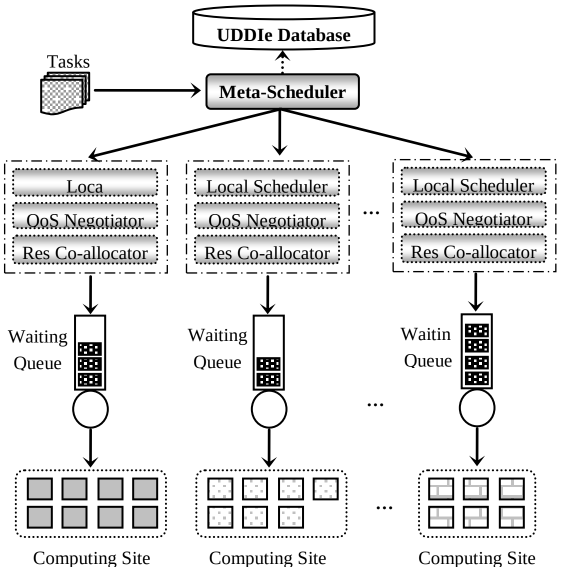


Figure 1. Co-scheduling Model in Multi-Cluster Grid

Given the grid system has N computing sites, noted as $(CE_1,\ldots,CE_N)$. Similar to Berten's study in [14], we use M/M/C queueing system to describe the working model of a certain computing site.

Definition 1. Computing site CE_i is characterized as a three-tuple $<\lambda_i,\mu_i,c_i>$, where λ_i represents the average arrival interval between two tasks, μ_i represents the average service time, and c_i is the quantity of computing resources in CE_i.

Definition 2. A task is characterized a two-tuple $<\mathbf{R},d>$, where vector $\mathbf{R}=<r_1,\ldots,r_m>$ is the resource requirements of each sub-tasks, and d is the deadline requirement of the task.

Definition 3. For a task $<\mathbf{R},d>$, a co-allocation scheme is the mapping of resource requirements to computing sites, noted as $S:\mathbf{R}\times\{1,\ldots,N\}\rightarrow\{0,1\}$.

Definition 4. $P(S,d)$ represents the probability of deadline-guarantee for task $<\mathbf{R},d>$ if using scheme S to co-allocate resources.

The core of this paper is to find a method to calculate $P(S,d)$ for both PSA_NRR and PSA_RR. Furthermore, based on $P(S,d)$ we developed a deadline-guarantee-enhanced hybrid-policy co-allocation model, which takes advantage of existing co-allocation policies to generate multiple co-allocation schemes and select the scheme that can provide optimal deadline-guarantee for applications.

3.2. Deadline Guarantee Evaluation

As mentioned in definition 1, computing site CE_i is modeled as a M/M/C queueing system and characterized by three-tuple $<\lambda_i, \mu_i, c_i>$. According to queueing theory [20], the arrival of tasks on CE_i is a Possion process with mean value λ_i, and the service time follows exponential distribution with a mean value μ_i. So, the load intensive of CE_i can be noted as $\rho_i = \lambda_i / (c_i \cdot \mu_i)$.

Theorem 1. If computing site CE_i is modeled as M/M/c_i queueing system, the probability that the sub-tasks sent on CE_i can be completed before deadline d is as following

$$\Pr\{\boldsymbol{\omega} \le d\} = \sum_{n=0}^{c_i} \delta \cdot \frac{(\rho_i \cdot c_i)^n}{n!} + \sum_{k=1}^{c_i \cdot \mu_i \cdot d - 1} \delta \cdot \frac{\rho_i^{k+c_i} \cdot c_i^{c_i}}{c_i!}$$

where $\delta = \left[\sum_{n=1}^{c_i} \frac{(\rho_i \cdot c_i)^n}{n!} + \frac{(\rho_i \cdot c_i)^{c_i}}{c_i} \frac{1}{1-\rho_i} \right]^{-1}$, $\boldsymbol{\omega}$ is the random variable representing the actual completing time of the sub-task, d is the deadline relative to the sub-task's arrival time.

Proof. Let $\boldsymbol{\psi}$ be a random variable representing the number of waiting sub-tasks in CE_i. According to queueing theory, the probability that there are k waiting sub-tasks in CE_i is

$$\Pr\{\boldsymbol{\psi} = k\} = \begin{cases} \delta \cdot \dfrac{\rho_i^{k+c_i} \cdot c_i^{c_i}}{c_i!}, & k > 0 \\ \sum_{n=0}^{c_i} \delta \cdot \dfrac{(\rho_i \cdot c_i)^n}{n!}, & k = 0 \end{cases} \tag{1}$$

$$\text{where } \delta = \left[\sum_{n=1}^{c_i} \frac{(\rho_i \cdot c_i)^n}{n!} + \frac{(\rho_i \cdot c_i)^{c_i}}{c_i} \frac{1}{1-\rho_i} \right]^{-1} \tag{2}$$

For M/M/c_i queueing system, the service rate is $c_i \mu_i$, which means the amount of sub-tasks that CE_i can complete in a unit time is $c_i \mu_i$. So, the amount of sub-tasks that CE_i can complete in period d is $c_i \mu_i d$. Therefore, the probability that CE_i can guarantee a sub-task's deadline is equal to the probability that the number of waiting sub-tasks in CE_i is not more than $c_i \cdot \mu_i \cdot d - 1$. That is

$$\Pr\{\boldsymbol{\omega} \le d\} = \Pr\{\boldsymbol{\psi} \le c_i \cdot \mu_i \cdot d - 1\} \tag{3}$$

By (1)(2)(3), we can get that

$$\begin{aligned} \Pr\{\boldsymbol{\omega} \le d\} &= \Pr\{\boldsymbol{\psi} \le c_i \cdot \mu_i \cdot d - 1\} \\ &= \sum_{k=0}^{c_i \cdot \mu_i \cdot d - 1} \Pr\{\boldsymbol{\psi} = k\} \\ &= \sum_{n=0}^{c_i} \delta \cdot \frac{(\rho_i \cdot c_i)^n}{n!} + \sum_{k=1}^{c_i \cdot \mu_i \cdot d - 1} \delta \cdot \frac{\rho_i^{k+c_i} \cdot c_i^{c_i}}{c_i!} \end{aligned} \tag{4}$$

■

The theorem 1 gives the probability of deadline-guarantee to a sub-task. Based on theorem 1, we develop the algorithm to calculate the deadline-guarantee of any co-allocation scheme for tasks in the scenario of PSA_NRR or PSA_RR. The details of the algorithm are shown as following.

```
Algorithm 1: Calculate $P(S,d)$ for PSA_NRR and PSA_RR
Input:
  $S$ : co-allocation matrix
  $d$ : relative deadline of tasks
Output:
 $P(S,d)$
Begin
1. $P(S,d) := 1$
2. For $j = 1$ to $N$
3.   $WaitLength[j] := c_j \cdot \mu_j \cdot d - 1$
4. End For
5. For $i := 1$ to $m$
6.   If the task is PSA_NRR then
7.     For $j := 1$ to $N$
8.       If $S_{i,j} = 1$ Then
9.         $P_i^j := \sum_{n=0}^{c_j} \delta_j \cdot \frac{(\rho_j \cdot c_j)^n}{n!} + \sum_{k=1}^{WaitLength[j]} \delta_j \cdot \frac{\rho_j^{k+c_j} \cdot c_j^{c_j}}{c_j!}$
10.        $P(S,d) := P(S,d) \cdot P_i^j$
11.        $WaitLength[j] := WaitLength[j] - 1$
12.        Continue
13.      End If
14.    End For
15.  Else // the task is PSA_RR
16.    $Max\_P_i^j = 0$
17.    For $j := 1$ to $N$
18.      If $S_{i,j} = 1$ Then
19.        $P_i^j := \sum_{n=0}^{c_j} \delta_j \cdot \frac{(\rho_j \cdot c_j)^n}{n!} + \sum_{k=1}^{WaitLength[j]} \delta_j \cdot \frac{\rho_j^{k+c_j} \cdot c_j^{c_j}}{c_j!}$
20.        If $Max\_P_i^j < P_i^j$ Then $Max\_P_i^j := P_i^j$
21.        $WaitLength[j] := WaitLength[j] - 1$
22.      End If
23.    End For
24.    $P(S,d) := P(S,d) \cdot Max\_P_i^j$
25.  End If
26. End For
End
```

For a task, if one of its sub-tasks has been scheduled on a computing site, the following sub-tasks' deadline guarantee will be affected. To deal with this case, we use array *WaitLength* to amend the calculating process, and the amended formulae (4) is shown in step 9 and step 19 of the algorithm.

4. Multi-Scheme Co-Scheduling Framework

In [2], the authors points out that "availability, reliability, usability of grid resources are greatly effected by many factors. So, there is no general co-allocation policy that can deal with all kinds of cases". So, we propose and design the HPCM, which is based on existing policies and provide enhanced deadline-guarantee for user's tasks. The prototype of HPCM is shown in Figure 2.

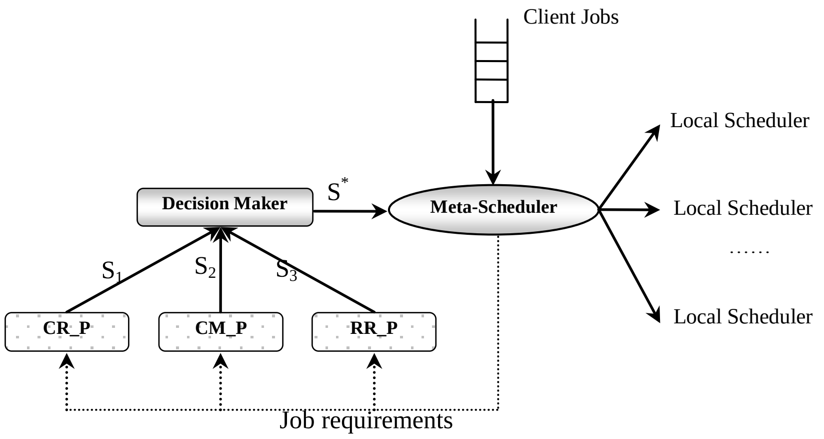


Figure 2. The Prototype of HPCM

In HPCM, the meta-scheduler makes multiple co-allocation schemes for the arrival tasks by using different existing policies. Then, *Decision Maker* evaluates the deadline-guarantee of these co-allocation schemes by using the algorithm shown in section 3.2. The scheme with the highest value of $P(S,d)$ is selected as the final co-allocation scheme for the task. Currently, three typical co-allocation policies are incorporated in HPCM. The descriptions of the three policies are as following:

- *Round Robin Policy* [21] (RR_P): The meta-scheduler assigns tasks to computing sites in turn. RR_P is easy to be implemented and effective to obtain load-balance in homogeneous systems.
- *Capability-based Random Policy* [14] (CR_P): The probability of selecting computing sites for a task is proportional to its processors' speed and number. This policy is driven by the intuition that more tasks should be assigned to more powerful computing sites.
- *Cluster Minimized Policy* [22] (CM_P): The meta-scheduler tries to assign a task to a set of computing sites, with arming at minimizing the size of the co-allocation set.

For the convenience of analysis and comparison in experiments, we add a set of control switches, by which we can turn on or turn off the *Decision Maker* in runtime and choose any of these three policies as the current co-allocation policy.

5. Experiments and Performance Comparison

5.1. Experimental Settings

We use GridSim [23], a distributed resource management and scheduling simulator, to evaluate the performance of HPCM. A multicluster computing grid model is constructed, which consists of twelve computing sites (CE_1~CE_12). The topology and deployment of the grid model is shown in Figure 4, which is referenced from grid test-bed DAS-2.

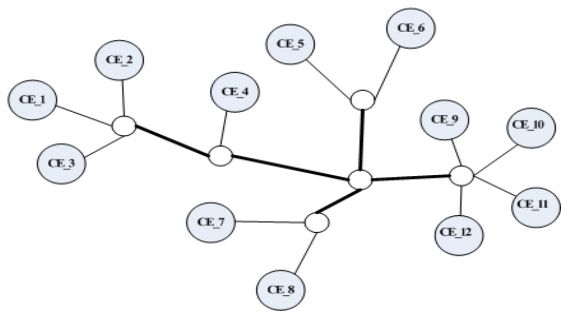


Figure 3. Computing Grid Model in Simulations

In simulations, the basic workload (tasks stream) is generated by using Lublin-Feitelson model [24], which is derived from the logs of real supercomputers. It consists of 10000 tasks, each is characterized by its arrival time, resource demands, and estimation of execution time. The resource demand of each task is enlarged by f times, where f is uniformly distributed in $[10,30]$, so as to simulate the co-allocation in large-scale grid environments. As the basic workload does not include deadline, we append each task with a deadline as $deadline = T_{arrival} + k \cdot T_{execution}$, where k is a random variable that uniformly distributed in $[5.5,10.5]$.

5.2 Deadline Miss Rate

In the first experiment, we compare the performance of HPCM with other three policies in term of deadline-miss rate. As shown in the study of [25], when Redundant Request technique (also called K-request) is used, K=2 or K=3 is more suitable than other higher value. Because the system's performance become almost the optimal when K=2 or K=3, and higher value of K will increase overload of the system. So, in our experiments, we only considerate the cases of K=2 and K=3. As to the implementation of Redundant Request, we adopt the Random Redundant technique that described in [25] in details.

This experiment is divided into three groups according to the three scenarios (PSA_NRR, PSA_RR(K=2), PSA_RR(K=3)). For each scenario, the experiment is conducted for four times. In the first three times, we turn off the *Decision Maker* in HPCM and use the RR_P, CR_P, CM_P as the default co-allocation policy. Only in the fourth time, we turn on the *Decision Maker*. All these four experiments use the same task stream as their workload. In order to make the system work in a stable state, the first 2000 tasks in the workload are scheduled by using RR_P policy. So, all the experimental results are based on the later 8000 tasks. The experimental results are shown in Figure 4.

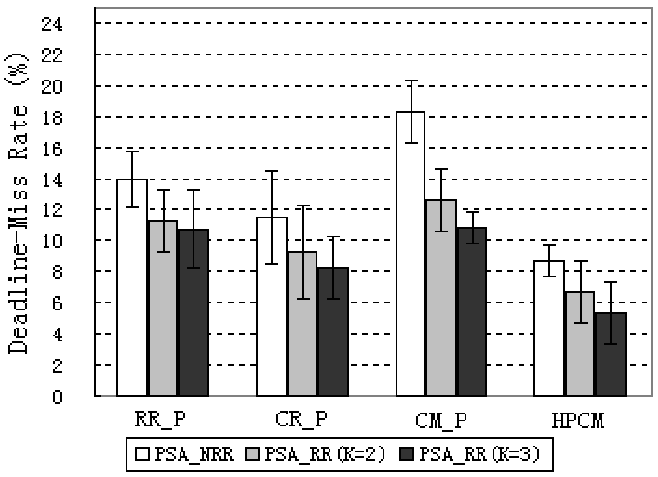


Figure 4. Deadline-Miss Rate of Different Policies

As we can see that, CR_P is more effective to reduce deadline-miss rate comparing with CM_P and RR_P, and CM_P performs worst in all experiments. After analyzing the simulative data in detail, we find that most of the tasks are scheduled on a few of computing sites (CE_3, CE_9, CE_10) when using CM_P. As the tasks in workload are all PSA type, CM_P can not fully exhibit its merits that reducing the communication overhead. It is the reason that CM_P's deadline-miss rate is the highest in all the co-allocation policies. On the other side, RR_P uniformly distributes the workload onto all computing sites, which in turn improve the PSA task's executive efficiency. During the experiment, we notice that most of the deadline-miss occur on these low-capability computing sites (i.e. CE_1, CE_6, CE_8) when using RR_P. On the contrary, CR_P gets over the disadvantages of both CM_P and RR_P by distributing the workload according to the capability of the computing sites, which is the main reason that CR_P is more effective than CM_P and RR_P. As shown in the Figure 4, when Redundant

Request is used the deadline-miss rate of the three policies are all reduced. Among them, CM_P's improvement is most significant. It is because that Redundant Request technique helps CM_P overcome its shortcoming of unbalancing-workload.

When HPCM is turn on, the HPCM evaluates the three policies' co-allocation scheme in term of deadline-guarantee. As shown in Fig. 4, the deadline-miss rate is significantly reduced when HPCM is used. For PSA_NRR, the deadline-miss rate of HPCM is 8.74%, which is reduced about 24% comparing even with CR_P; for PSA_RR(K=2), it is 6.68%; and for PSA_RR(K=3), it is only 5.32%. However, the results in Figure 4 only indicate the overall performance of HPCM. In order to examine the detailed working of HPCM, we record all the decisions made by HPCM during the experiment. In Table 1, the distribution of HPCM's decision is listed in details.

Table 1. Decision Distribution of HPCM

	Decision Counts (Deadline-Miss Counts)		
	RR_P	CR_P	CM_P
PSA_NRR	2253 (214)	4839 (357)	908 (128)
PSA_RR (K=2)	1855 (139)	4774 (332)	1371 (73)
PSA_RR (K=3)	2075 (152)	4116 (204)	1809 (64)

It can be seen that HPCM selects CR_P as the final co-allocation policy for over 50% tasks. When Redundant Request is used, the percentage of selecting CM_P is increased quickly, especially when K=3 the percentage of selecting CM_P is about twice as when Redundant Request is not used. The increasing of selecting CM_P indicates that CM_P's deadline-guarantee increases quickly when Redundant Request is used, and HPCM is adaptive to meet such situation. From the experimental results and above analysis, we can draw the conclusion that HPCM is an extendable co-allocation framework, which can fully take advantage of most of the existing co-allocation policies to meet the user's deadline requirement. Also, HPCM is suitable to those systems that using Redundant Request technique.

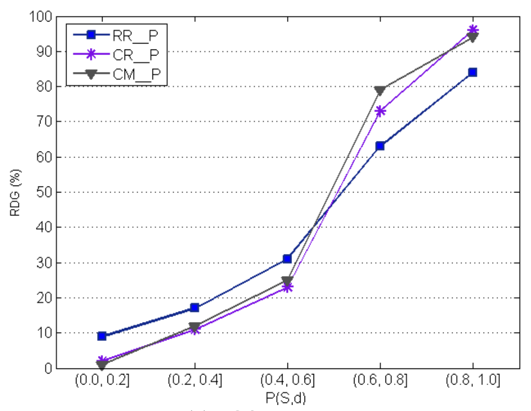
5.3. Evaluation on the Validity of Pr{S|d}

As noted above, HPCM uses the algorithm (shown in Section 3.2) to evaluate the co-allocation scheme's deadline-guarantee. An important issue in HPCM is the relationship between the value of $P(S,d)$ calculated by HPCM and the actual deadline-miss rate in practice. So, in this experiment we examine such relationship, from which we want to verify the validity of HPCM.

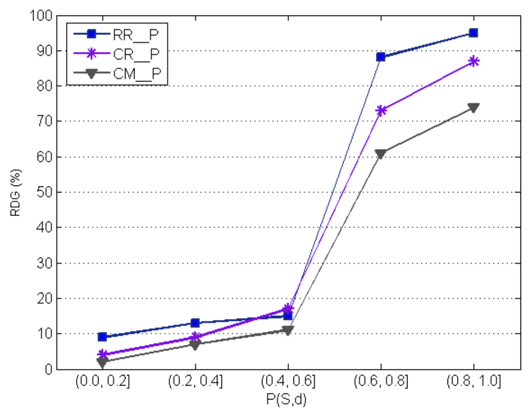
In this experiment, we turn off the Decision Maker and let the RR_P, CR_P, CM_P co-allocate resources for arriving tasks in turn. The workload is the same as the first experiment. In order to make the system work in a stable state, the first 1000 tasks in the workload are scheduled by using RR_P. Therefore, each policy will co-allocate resources for 3000 tasks. During the experiment, we calculate the value of $P(S,d)$ for each co-allocation scheme, and their execution results that whether the tasks are completed before their deadline requirements or not. To examine the validity of HPCM, we introduce a new metric, called *Rate of Deadline-Guarantee* (RDG), which is calculated as

$$RDG = \frac{\textit{the amount of tasks completed before deadline}}{\textit{total amount of tasks}}$$

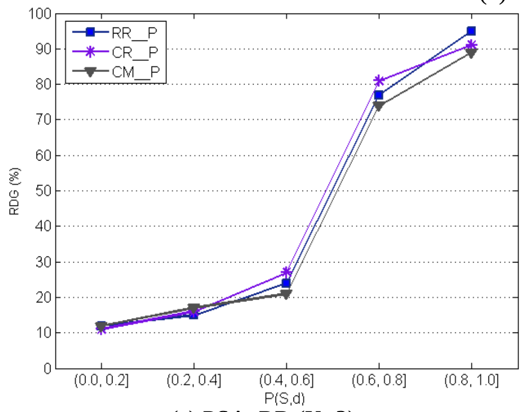
In order to show the relationship between the theoretical $P(S,d)$ and the actual RGD, we group the values of all $P(S,d)$ into five intervals as $(0,0.2]$, $(0.2,0.4]$, $(0.4,0.6]$, $(0.6,0.8]$, $(0.8,1.0]$. For each interval, we calculate the corresponding RDG that are shown in Figure 5.



(a) PSA_NRR



(b) PSA_RR (K=2)



(c) PSA_RR (K=3)

Figure 5. Rate of Deadline-Guarantee in Various Interval of $P(S,d)$

As show in Figure 5(a), when Redundant Request is not used, the RDG is very close to the theoretical value of $P(S,d)$ and the average differences between them is about 6%, especially for RR_P. When Redundant Request is used(shown in Figure 5(b)(c)), we notice that, when $P(S,d) < 0.6$ the values of $P(S,d)$ seem to underestimate the RDG; when $P(S,d) > 0.6$ the values of $P(S,d)$ seem to overestimate the RDG. To investigate such phenomenon, we analyze the detailed process of the experiment and find: (1) The co-allocation schemes with lower $P(S,d)$ tend to schedule most of the sub-tasks onto several high-performance computing sites, which frequently result in deadline missing; (2) Those schemes with higher $P(S,d)$ tend to send sub-tasks onto the computing sites with low-workload and high-performance, in such case tasks often can be completed before their deadline. Based on these findings, it is clear that using the co-allocation schemes with higher value of $P(S,d)$ not only can provide deadline-guarantee to tasks, but also can balance the workload between computing sites. When Redundant Request is used, we notice that the RDG of various policies tend to be the same. It is because that many schemes obtained from different policies are actually the same when using Redundant Request technique. Such similarity becomes very clear when using higher value of K. Also, the values of $P(S,d)$ reflect such similarity (as shown in Figure 5(c)). In our simulative setting, there are 12 computing sites. Assume that we set K=12, the differences between different policies will be vanished.

6. Conclusion

Co-allocation in computational grid is an important issue with increasing concerns. Although many architectures and policies have been developed for supporting efficient co-allocation, however, co-allocation with constraints to deadline still remains unsolved. In this work, we address this issue by presenting a novel Hybrid-Policy Co-allocation Model (HPCM), which takes advantage of existing co-allocation policies to provide optimal deadline-guarantee for applications. In simulations, we conducted extensive experiments to investigate the performance and validity of the proposed model. From the experimental results and the analysis, we find that the HPCM is an extendable co-allocation framework to meet user's deadline requirement. At present, we have only implemented three kinds of typical co-allocation policies in the prototype of HPCM. In the future, we will incorporate more policies in it. Also, the failure of resources in grid environment is not considered in our HPCM, which have great influences on the deadline-guarantee for applications. So we plan to take more efforts on the reliability of co-allocation schemes in our next step.

Acknowledgements

This work is supported by the Hunan Provincial Natural Science Foundation of China (14JJ7071).

References

[1] J. H. Park, L. T. Yang and J. Chen, "Research trends in cloud, cluster and grid computing," Cluster Computing-the Journal of Networks Software Tools and Applications, vol. 16, no. 3, **(2013)**, pp. 335-337.

[2] N. Muthuvelu, C. Vecchiola and I. Chai, "Task granularity policies for deploying bag-of-task applications on global grids", Future Generation Computer Systems, vol. 29, no. 1, **(2013)**, pp. 170-181.

[3] A. Mamat, Y. Lu and J. Deogun, "Scheduling real-time divisible loads with advance reservations", Real-Time Systems, vol. 48, no. 3, **(2012)**, pp. 264-293.

[4] M. J. Lewis and A. J. Ferrari, "Support for Extensibility and Site Autonomy in the Legion Grid System Object Model", Journal of Parallel and Distributed Computing, vol. 63, no. 5, **(2003)**, pp. 525-538.

[5] R. Wolski, J. Brevik and G. Obertelli, "Writing programs that run EveryWare on the computational grid", IEEE Transactions on Parallel and Distributed Systems, vol. 12, no. 10, **(2001)**, pp. 1066-1080.

[6] A. L. Chervenak, R. Schuler and M. Ripeanu, "The Globus Replica Location Service: Design and Experience", IEEE Transactions on Parallel and Distributed Systems, vol. 20, no. 9, **(2009)** September, pp. 1260-1272.

[7] W. Leinberger, G. Karypis and V. Kumar, "Job scheduling in the presence of multiple resource requirements", Proceedings of ACM/IEEE Conference on Supercomputing, **(1999)**, pp. 108-117.

[8] H. H. Mohamed and D. Epema, "An evaluation of the close-to-files processor and data co-allocation policy in multiclusters", Proceedings of Int'l Conference on Cluster Computing, **(2004)**, pp. 287-298.

[9] L. He and S. A. Jarvis, "Allocating non-real-time and soft real-time jobs in multiclusters", IEEE Transactions on Parallel and Distributed Systems, vol. 17, no. 2, **(2008)**, pp. 99-112.

[10] A. Bucur and D. Epema. "Scheduling policies for processor coallocation in multicluster System", IEEE Transactions on Parallel and Distributed Systems, vol. 18, no. 7, **(2007)**, pp. 958-962.

[11] M. Finger, G. C. Bezerra, and D. R. Conde, "Resource use pattern analysis for predicting resource availability in opportunistic grids," Concurrency and Computation-Practice & Experience, vol. 22, no. 3, pp. 295-313, 2010.

[12] C.-M. Wang, H.-M. Chen, C.-C. Hsu, "Dynamic resource selection heuristics for a non-reserved bidding-based Grid environment," Future Generation Computer Systems, vol. 26, no. 2, pp. 183-197, 2010.

[13] C.-T. Yang, F.-Y. Leu and S.-Y. Chen, "Resource brokering using a multi-site resource allocation strategy for computational grids", Concurrency and Computation-Practice & Experience, vol. 23, no. 6, **(2011)**, pp. 573-594.

[14] V. Berten, J. Goossens and E. Jeannot, "On the distribution of sequential jobs in random brokering for heterogeneous computational grids", IEEE Transactions on Parallel and Distributed Systems, vol. 17, no. 2, **(2007)**, pp. 113-124.

[15] K. Kurowski, A. Oleksiak and W. Piatek, "Hierarchical scheduling strategies for parallel tasks and advance reservations in grids", Journal of Scheduling, vol. 16, no. 4, **(2013)**, pp. 349-368.

[16] Z. C. Papazachos and H. D. Karatza, "Gang scheduling in multi-core clusters implementing migrations", Future Generation Computer Systems, vol. 27, no. 8, **(2011)**, pp. 1153-1165.

[17] I. A. Moschakis, and H. D. Karatza, "Evaluation of gang scheduling performance and cost in a cloud computing system", Journal of Supercomputing, vol. 59, no. 2, **(2012)**, pp. 975-992.

[18] H. E. Bal. "The distributed ASCI supercomputer project," ACM Operating Systems Review, vol. 34, no. 4, pp. 76-96.

[19] M. Wu, X. H. Sun and Y. Chen, “QoS oriented resource reservation in shared environments”, Proceedings of Int’l Symposium on Cluster Computing and the Grid, **(2006)**, pp. 601-608.
[20] D. Gross and C. M. Harris, “Fundamentals of Queuing Theory”, USA: John Wiley and Sons, **(1998)**.
[21] G. Koole and R. Righter, “Resource allocation in grid computing”, Journal of Scheduling, vol. 11, no. 3, **(2008)**, pp. 163-173.
[22] C. Li and L. Li, “Cross-layer optimization policy for QoS scheduling in computational grid,” Journal of Network and Computer Applications, vol. 31, no. 3, **(2008)**, pp. 258-284.
[23] Y. Hao, G. Liu, and N. Wen, “An enhanced load balancing mechanism based on deadline control on GridSim”, Future Generation Computer Systems, vol. 28, no. 4, **(2012)**, pp. 657-665.
[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. 63, no. 11, **(2003)**, pp. 1105-1122.
[25] H. Casanova. “Benefits and drawbacks of redundant batch requests,” Journal of Grid Computing, vol. 5, no. 2, **(2007)**, pp. 235-250.

Authors

Dongbo Liu, received his doctor degree in Hunan University in 2012. Now he works in Hunan Institute of Engineering as an associate professor. His research interests include distributed system, cloud computing, high-performance application. He is now a member of CCF in China, and worked as Senior Engineer in HP High-performance Lab of Hunan Institute of Engineering.

Ning Han, received his bachelor degree in Beijing University of Science and Technology, and now persuading master degree in Xiangtan University. Currently, he works in the HP High Performance Lab of Hunan Institute of Engineering as a senior networking engineer. His research interests include complex networking deployment, distributed computing, information security technology, fault-tolerance in distributed systems.