Policy-based Resource Allocation in Hierarchical Virtual Organizations for Global Grids

Kyong Hoon Kim and Rajkumar Buyya
Grid Computing and Distributed Systems (GRIDS) Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
E-mail: {jysh, raj}@csse.unimelb.edu.au

Abstract

In recent years, many studies have been conducted on Grid computing, in which users and resource providers organize various Virtual Organizations (VOs) to share resources and services. A VO organizes other sub-VOs for the purpose of achieving the VO goal, which forms the hierarchical VO environment. In this paper, we model and formalize the resource allocation problem in hierarchical VOs. Resource providers and VOs agree upon the VO resource sharing policy, such as resource sharing amount and resource usage cost for VOs. We provide the resource allocation scheme of a VO resource broker to minimize the total cost in order to meet a user's job deadline. In addition, we deal with several cost adjustment methods in resource providers to utilize their resources efficiently in hierarchical VOs.

1. Introduction

The rapid growth of computer and network technologies has enabled global computing with plenty of resources which are heterogeneous and distributed geographically. The Grid has started from the realization of scientific computations over geographically distributed systems and has been an emerging technology in recent years [1, 2]. Many studies on Grid computing have been conducted, such as resource allocation, resource management, security, and Web Services [2, 3, 4, 5].

A Virtual Organization (VO) in the Grid is defined as a set of individuals and institutions forming an ad-hoc partnership to solve a common problem by sharing resources [1]. Recent research has focused on VO-based services, including VO formation, operation, and resource allocation. Thus, large-scale Grid research projects [6, 7] provide VO services and organize various VOs to utilize distributed resources efficiently. In VO-enabled Grid environments, the VO-wide resource allocation problem becomes an emerging research topic, which enables a user to access several resources throughout VOs. Much research [8, 9, 10] has been done on policy-based resource allocation in VOs. The resource broker allocates resources to a job according to VO policy, such as the amount of resource share.

As the number of VOs increases in the Grid, efficient VO management is required. The VO organizes its own sub-VOs for the purpose of achieving the VO goal, which forms the hierarchical VO environment. This paper deals with resource allocation problem in hierarchical VOs. Another important issue in Grid computing is economy-based resource allocation [11], which minimizes the resource usage cost of a user. We also include the resource usage cost in VO policy model.

The main contributions of this paper are as follows: (i) to model hierarchical VO environments for global Grids and formalize the resource allocation problem; (ii) to provide a VO-wide resource allocation scheme to minimize cost in order to meet a user's job; and (iii) to propose possible cost adjustment methods to resource providers in hierarchical VOs.

The remainder of this paper is organized as follows. In Section 2, we present related work on VO-wide resource allocation in the Grid. We define the hierarchical VO system model and formalize the resource allocation problem in Section 3. Section 4 discusses the proposed resource allocation framework including the resource allocation scheme in the resource broker and the cost adjustment scheme in the resource site. We show simulation results in Section 5 and conclude the paper in Section 6.

2. Related Work

Recent large-scale Grid projects include VO facilities to federate various distributed resources. The OSG (Open Science Grid) [6] provides a Grid infrastructure for large-scale science applications and resources sharing throughout VOs. The EGEE (Enabling Grids for E-SciencE) [7] also organizes many VOs and shares resources to manage resources efficiently. As the Grid computing expands world-wide, the VO facility is required to integrate various resources. The VOMS (Virtual Organization Membership Service) [12] is an authentication service supporting VOs in Globus toolkit. Most of VO services are built based on VOMS since authentication and authorization is a basic service for supporting VO. Resource providers want only authorized VO users to use their resources.

Recent research on Grid computing has focused on the VO-wide scheduling and resource allocation based on VO polices. In [8], they introduce a new framework for policy based scheduling as a part of SPINIX scheduling system. The scheduling strategy in the framework adjusts resource usage accounts or request priorities for efficient resource usage management. Dumitrescu and Foster [9] propose a usage policy-based scheduling in VOs and evaluate both aggregate resource utilization and aggregate response time. The evaluated usage polices are fixed limit, extensible-limit, and commitment-limit, in which the limit is a fraction of resource in a resource site provided to a specific VO. The commitment-limit policy defines two upper limit: an epoch limit for the average resource utilization limit for a long time and a burst limit for a short time period limit. They propose a prototype resource broker GRUBER [13] for resource usage SLA specification and enforcement in a Grid environment.

Elmroth and Gardfjall [10] have presented a decentralized architecture for a Grid-wide fair scheduling system, where each local scheduler enforces Grid-wide hierarchical share policies using a global resource usage data. The policy engine calculates a fairshare priority factor for a job to support the Grid-wide share policy. Sulistio and Buyya [14] proposed a time optimization algorithm in auction-based proportional share systems with multiple VOs, in which a user broker periodically adjusts a bidding price in order to meet the deadline and minimize the cost. Norman, et. al. [15] developed a model of VO management that operates in complex electronic commerce scenarios. They suggest how to organize a VO for satisfying a user's various service requests. A VO in [15] is defined as a unit of economic services among users and service providers.

Previous research has investigated various policy attributes, such as time [8, 9], resource usage [9, 10], share [10], and cost [14]. Time attribute defines the period of access of a VO user. Resource usage refers to how much a VO user can use a resource in terms of the number of processors [13] or the percentile [9, 10]. Share policy enforces the sharing of resources between projects, groups, and users in a VO. The cost policy defines the resource usage cost for a VO user.

Although the policy models in [8, 10] are based on a VO hierarchy, they assume that resource providers only define the resource sharing of root VOs in VO policy trees, which is called local policy in [10]. All other VOs in policy trees follow the share specified in the policy tree. However, resource providers can provide their resource to any VOs in a VO hierarchy, as well as root VOs. This motivates our research so that we investigate resource allocation in such a case.

Another research motivation is the consideration of resource cost policy in VO-wide resource allocation. In [14], they define two different static costs for a VO and non-VO resource usage. However, in an economy-based Grid environment [11], resource providers adjust their resource costs dynamically according to their current status, such as system utilization. Thus, this paper considers the resource usage cost for a VO and introduces cost adjustment schemes for an efficient resource use. The resource allocation scheme also considers the minimization of the total cost for running a job.

3. Hierarchical Virtual Organizations

3.1. Hierarchical VO Environment

As many VOs are organized in the Grid, it is necessary to federate VOs or share services between VOs. A VO can also divide itself into several sub-VOs for the efficient management. Thus, we define and view a VO as a set of users, resource providers, and sub-VOs, as in [6]. Sub-VOs have similar aims as the VO.

Figure 1 shows an example of hierarchical VOs. VO-A consists of user U1, resource R1, and two sub-VOs (VO-A1 and VO-A2). VO-B is composed of three sub-VOs. A sub-VO can include other sub-VOs, as in VO-A1. Resource

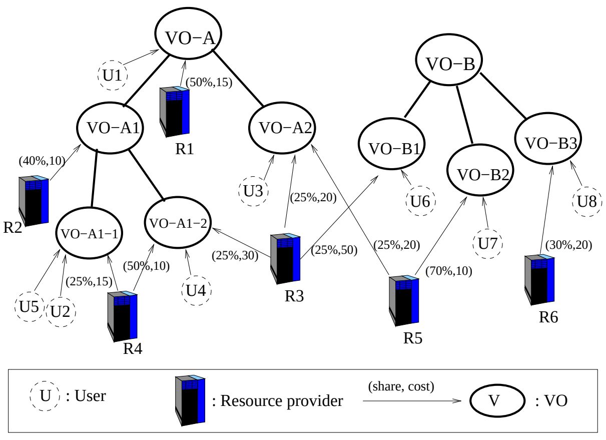


Figure 1. An example of hierarchical VOs

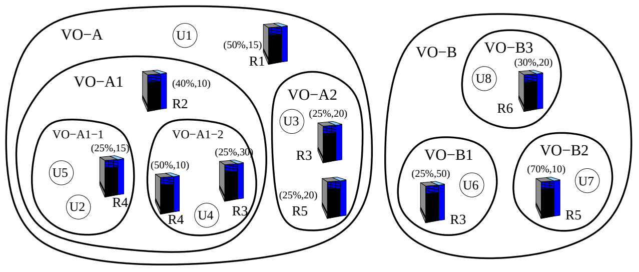


Figure 2. Container view of VOs in Figure 1

providers can share their resources to several VOs. For example, R3, R4, and R5 in Figure 1 provide their resources to multiple VOs. Figure 2 shows another way of viewing the hierarchical VOs in Figure 1.

Members in a VO share resources under the same policy, such as resource usage. A VO policy in a hierarchical VO applies to all the users in sub-VOs in the hierarchy. Thus, a user belongs to multiple VOs and can access the resources in these VOs. A user runs a job for his or her VO's goal which is a part of its ancestor VOs. Therefore, a user can use the resources of ancestor VOs. For example, user U2 in Figure 1 can use R4 of VO-A1-1, R2 of VO-A1, and R1 of VO-A.

3.2. System Model

3.2.1. VO model

The system components in global Grids are users, resource providers, and VOs. A *user* is an end-entity who submits jobs to the Grid and runs the jobs using the resources in VOs. A *resource provider* shares its resources to users in the Grid, especially to users in VOs that each resource provider has joined in. A *VO* is an organization of users, resource providers, and sub-VOs to meet the goal of the VO. Thus, we define the global Grids as $G = (U, R, V)$, where U is a set of users, R is a set of resource providers, and V is a set of VOs in the Grids.

We denote each set of users, resource providers, and sub-VOs in a VO v as U_v, R_v, and V_v, respectively, so that a VO v is defined by (U_v, R_v, V_v). Table 1 shows the VO components of each VO in Figure 1. In hierarchical VOs, we additionally define the following terminologies.

- *Parent VO:* If a VO v is one of sub-VOs of v', ($v \in V_{v'}$), we call v' a *parent* VO of v. We denote it as *parent (v).*
- *Ancestor VOs:* For a given VO v, all the VOs in the path from v to the root in its VO tree are called *ancestor* VOs of v. We denote it as *ancestor (v).*

Table 1. System components in Figure 1

VO (v)	VO Components		
	Users (U_v)	*Resource providers (R_v)*	*Sub-VOs (V_v)*
VO-A	{U1}	{R1}	{VO-A1, VO-A2}
VO-A1	∅	{R2}	{VO-A1-1, VO-A1-2}
VO-A2	{U3}	{R3}	∅
VO-A1-1	{U2, U5}	{R4}	∅
VO-A1-2	{U4}	{R3, R4}	∅
VO-B	∅	∅	{VO-B1, VO-B2, VO-B3}
VO-B1	{U6}	{R3}	∅
VO-B2	{U7}	{R5}	∅
VO-B3	{U8}	{R6}	∅

- *Root VO:* If a VO v has no parent VO, it is called a *root* VO.
- *Leaf VO:* If a VO v has no sub-VOs ($V_v = \varnothing$), it is called a *leaf* VO.
- *Intermediate VO:* If a VO is neither root nor leaf, it is called an *intermediate VO.*

Let us examine hierarchical VOs in Figure 1 as an example. Figure 1 has tow root VOs: VO-A and VO-B, one intermediate VO: VO-A1, and six leaf VOs: VO-A2, VO-B1, VO-B2, VO-B3, VO-A1-1, and VO-A1-2. The ancestor VOs of VO-A1-1 are VO-A1 and VO-A.

3.2.2. Policy model

In this paper, we consider VO polices between resource providers and their VOs in two aspects: resource share and resource cost.

- *Resource share policy:* This policy implies how much of its resource a resource provider shares in a VO. The current resource share amount is denoted as *share (r, v)*, where $r \in R_v$. The maximum amount of resource share amount is denoted as $share_{max}$ *(r, v).*
- *Resource cost policy:* This policy defines a resource usage cost in a VO user. The current resource usage cost is denoted as *cost (r, v)*, where $r \in R_v$. The maximum cost of resource usage is denoted as $cost_{max}$ *(r, v).*

The resource share amount indicates the percentile of the total resource in a resource provider. It has different meanings according to the resource provider's sharing policy. For the space-shared scheduling policy, the share amount implicitly implies the number processors provided to VOs. For the time-share policy, it denotes the proportion in the total processing power of the resource provider shared to VOs. Our simulations in Section 5 use the time-shared scheduling policy.

Each resource provider *r* agrees that users can use the resource up to the maximum amount of $share_{max}(r, v)$ for the maximum cost of $cost_{max}(r, v)$ for a VO *v*. For example, R5 in Figure 1 provides 25% of resource to VO-A2 for the cost of 20 and 70% to VO-B2 for the cost of 10. Table 2 lists VO policies of resource providers in Figure 1.

Table 2. VO policies in Figure 1

Resource provider (*r*)	VO policy		
	VO (*v*)	share	cost
R1	VO-A	50%	15
R2	VO-A1	40%	10
R3	VO-A2 VO-A1-2 VO-B	25% 25% 25%	20 30 50
R4	VO-A1-1 VO-A1-2	25% 50%	15 10
R5	VO-A2 VO-B2	25% 70%	20 10
R6	VO-B3	30%	20

The current available resource amount to each VO, *share (r, v)*, changes dynamically, as VO users run jobs at the resource. The current resource usage cost is also changed in the run-time. It is possible to provide dynamic cost adjustment according to the current resource utilization. We investigate possible cost adjustment schemes in Section 4.3.

3.2.3. Job model

A job in this paper is considered to be a bag-of-tasks application [16], which consists of multiple independent tasks with no communication among each others. In order to obtain the job's result, these tasks should be completed. In addition, we specify the deadline of a job as QoS parameter, so that the job execution must be finished before the deadline.

Thus, a user's job is defined as $(p, \{l_1, l_2, \ldots, l_p\}, d)$, where *p* is the number of sub-tasks, l_i is the number of instructions of the *i*-th task in Million Instructions (MIs), and *d* is the deadline. The execution time of a task of length l_i varies according to the processor performance of the resource on which the task is run. Since the execution time is easily obtained from the task length on a resource provider, we use the task length as a task specification instead of the execution time. We also assume that the number of instructions of each task is known in advance.

3.3 Formalizing Resource Allocation Problem

Users in a VO *v* use resources in R_v in order to accomplish the VO purpose. In addition, since users also belong to their ancestor VOs, they can use resources in the ancestor VOs of *v* as well. We define the set of all resources in the Grid for a user *u* to access as R_u^G.

$$R_u^G = R_v \cup \left\{ \bigcup_{w \in ancestor(v)} R_w \right\} \quad \text{, where } u \in U_v$$

Now, we formalize a resource allocation problem in hierarchical VO environments. The objective of the resource allocation is to minimize the total resource cost of a user job in order to meet the job deadline. The VO-wide resource allocation problem is defined as follows.

Definition 1. (VO-wide Resource Allocation Problem) *For a given job $J = (p, \{l_1, l_2, \ldots, l_p\}, d)$ of a user u, the resource allocation problem is defined as mapping of i-th task of l_i length to a resource $r_i \in R_u^G$ as to minimize the total cost $\sum_{i=1}^{p} cost(r_i, v_i)$, subject to the condition that the total resource usage in r_i is up to share (r_i, v_i) and each task completes before the deadline d.*

4. VO-wide Resource Allocation

4.1. VO-wide Resource Allocation Framework

The proposed VO-wide resource allocation framework uses a cooperative VO resource broker system. Each VO has a resource broker for the VO users and resource providers. The VO resource broker manages VO policies in the VO and plays a role in allocating jobs submitted by the VO users. It also provides VO policy information to other VO resource brokers. Users and resource providers know locations or service contact points of their VO resource brokers. Figure 3 shows system components and illustrates resource allocation procedures.

(1) *Submitting jobs*. When a user submits a job, he or she specifies the VO information as well as the job. The user attaches the VO attribute policy, such as the attribute certificate in VOMS [12]. The job with VO policy is submitted to the VO resource broker (VO-RB). Then, the VO resource broker checks the validity of the submitted job with the VO policy engine.

(2) *Gathering resource sharing information*. In order to provide the best resources to the user, the broker gathers resource sharing information from the

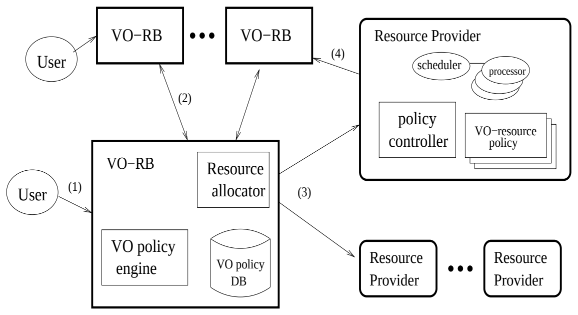


Figure 3. The VO-wide resource allocation framework

ancestor VOs in the VO policy tree. The user can access the resources of the ancestor VOs because the job aims at not only the VO itself but also the ancestor VOs.

(3) *Allocating resources*. The VO resource broker allocates resources to the job based on the resource sharing information aggregated from other VOs. Tasks of the job can be divided into several resource providers according to loads in resource providers. The task acceptance is accomplished by the local scheduler in each resource provider.

(4) *Updating sharing polices*. If each resource provider receives a job from the broker, it first validates the job in accordance with the VO policy. For example, the user's VO should be one of the resource provider's VOs or their child VOs. Then, it schedules the job with the local scheduler. The resource provider updates the changed polices to the corresponding VO resource broker.

4.2. Resource Allocation Scheme

The VO resource broker aims to minimize the total cost for a user's job in order to meet the job deadline under VO resource policies. Figure 4 shows the pseudo resource allocation algorithm of the VO resource broker.

First, the VO resource broker queries the current available resources from the ancestor VO resource brokers in the VO policy tree. It constructs the set of resources, R_u^G, for the job and sorts the resources in the increasing order of the cost (line 1 ~ 3).

The allocation scheme is to select the resource with the minimum cost first. The function **Submit** in line 7 of Figure 4 sends the job to the selected resource r and returns the number of tasks allocated to that resource. Each resource provider has its own local scheduler, so that it schedules unallocated tasks of the submitted job as long as the used

```
    Algorithm VO_wide_Resource_Allocation ($J$, $v$)
     /*  - $J = (p, \{l_1, \ldots, l_p\}, d)$ : a job
         - $v$ : the VO
     */
1:  Get resource sharing information from ancestor VOs.
2:  Construct $R_u^G = \{r \mid r \in R_v$ or $r \in R_{ancesotr(v)}\}$.
3:  Sort $R_u^G$ in the increasing order of $cost\ (r,v)$.
4:  $task\_index \leftarrow 0$;
5:  for $k$ from 1 to $|R_u^G|$ do
6:      Get the $k$-th $r$ in $R_u^G$.
7:      $num\_alloc \leftarrow$ Submit ($J, task\_index, r$);
8:      $task\_index \leftarrow task\_index + num\_alloc$;
9:      if ($task\_index == p$) return admit;
10: endfor
11: for $k$ from 1 to $task\_index$ do
12:     Cancel the $k$-th task.
13: return reject;
```

Figure 4. VO-wide resource allocation scheme

share amount of the VO does not exceed the total share allowed to that VO. The local scheduler accepts the sub-tasks only if it can meet their deadlines.

If all p sub-tasks are successfully allocated, the algorithm ends and the job is accepted (line 9). However, if there is no sufficient resource to run the job, it cancels all the previously allocated sub-tasks and rejects the job (line 11 ~ l3). The user can submit the rejected job later again, or the resource broker can manage the waiting queue for those rejected jobs.

4.3. Cost Adjustment Policies

One important issue is how a resource provider adjusts VO share policy dynamically to utilize the resource efficiently. The resource provider may increase or decrease the amount of shared resource according to the system load. Users can access up to the amount of $share_{max}\ (r, v)$ on the resource share policy.

In case of the resource cost, a resource provider is in charge of adjusting the cost up to $cost_{max}\ (r, v)$. We consider three cost adjustment schemes for a given maximum cost policy: Static-Cost, Dynamic-VO-cost, and Dynamic-Load-Cost.

(i) *Static-Cost*: It fixes the resource cost as $cost\ (r, v)$ between zero and $cost_{max}\ (r, v)$, regardless of the system status.

(ii) *Dynamic-VO-Cost*: This policy changes the current cost according to the resource usage amount of a VO. Since $share\ (r, v)$ is the current available resource amount, the share currently used by the VO *v* is defined by $share_{max}(r, v) - share(r, v)$. Then, the resource cost for a VO *v* is determined by the following:

$$cost(r,v) = cost_{\max}(r,v) * \frac{share_{\max}(r,v) - share(r,v)}{share_{\max}(r,v)}$$

(iii) *Dynamic-Load-Cost*: This policy adjusts the current cost according to the current resource load, as well as the VO resource usage. If the current total resource load is denoted as $load_r$, then the cost of a VO *v* is defined as:

$$cost(r,v) = cost_{\max}(r,v) * \frac{share_{\max}(r,v) - share(r,v)}{share_{\max}(r,v)} * load_r$$

5. Simulation Results

In this section, we simulate the proposed resource allocation schemes using the GridSim toolkit [17, 18]. Figure 5 shows the simulated hierarchical VO environment with five VOs, six resource providers, and one user in each VO. The resource characteristics of six resource providers are shown in Table 3. As shown in Figure 5, R1500 and R1000-1 provide all the resources to VO1 and VO2 respectively. Other resource providers contribute their resources to their VOs evenly. We assume that each VO user continuously generates and submits jobs for the VO.

Each user's jobs are generated by the Poisson distribution with the inter-arrival time of 5 minutes. The number of tasks in each job is selected randomly between 2 and 32. Each job length is in the range from 100,000 MIPS to 1,000,000 MIPS. The deadline is selected from 20 % to 100 % more than the average execution time. The number of total submitted jobs of each user is 1000.

First, we assume all the resource providers use Dynamic-VO-Cost as its cost adjustment scheme with the maximum

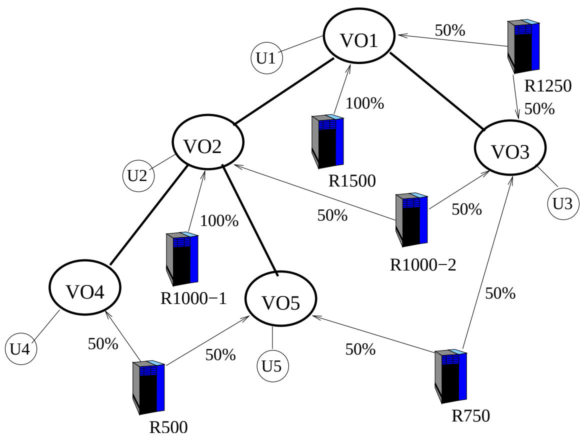


Figure 5. The simulated VO environment

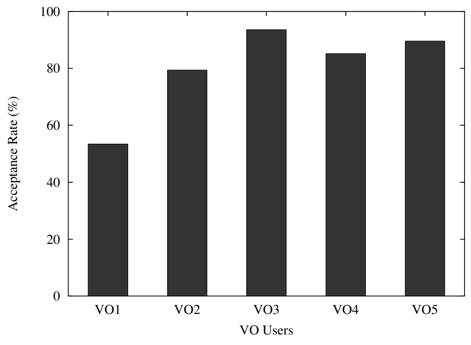


Figure 6. Job acceptance rates

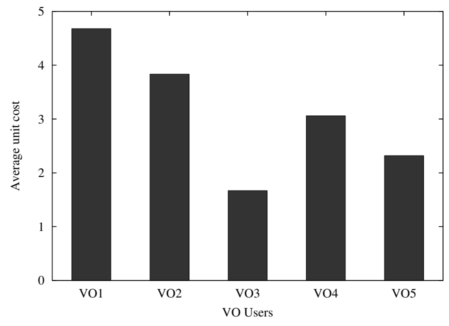
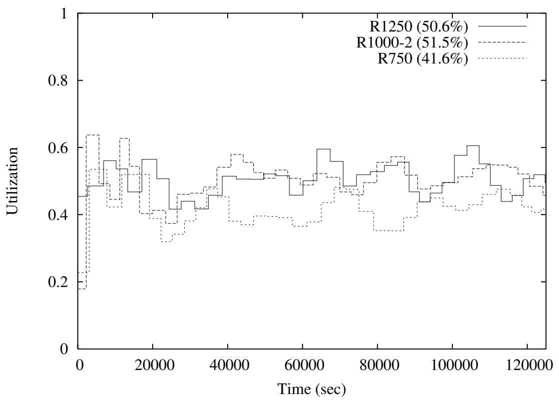


Figure 7. Average unit costs

cost of 10. The resource broker accepts only the jobs that can meet their deadlines; otherwise, it rejects them. Although these rejected jobs can be resubmitted later by users, our simulations do not consider these rejected jobs. Figure 6 shows the job acceptance rates of each VO user, which indicate how many jobs are completed before the deadlines. Since VO3 user can access more resource including 50% of R1250, 50% of R1000-2, 50% of R750 and shares R1500 through VO1, its job acceptance rate is the highest. Users VO4 and VO5 also show high job acceptance rates because they can use resources in ancestor

Table 3. Resource characteristics

Resource provider (*r*)	Resource characteristics	
	Processor performance (MIPS)	Number of processors
R1500	1500	20
R1250	1250	20
R1000-1	1000	20
R1000-2	1000	20
R750	750	20
R500	500	20



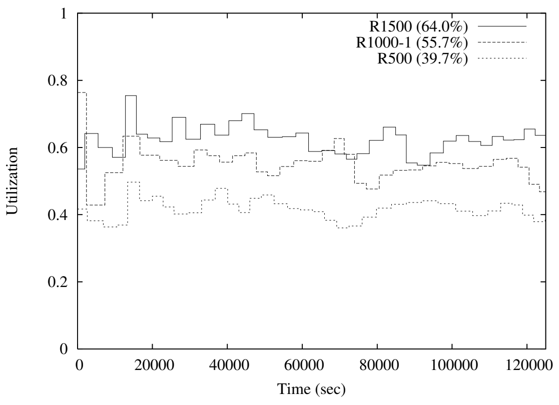
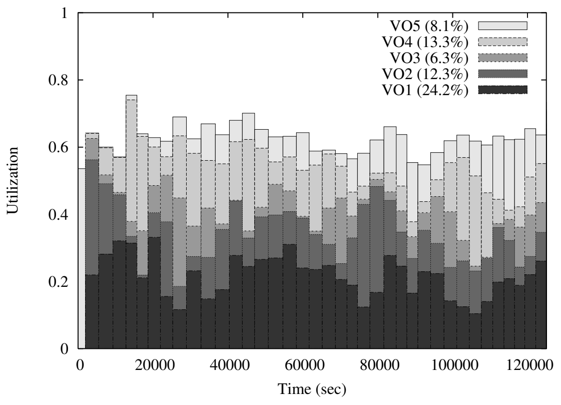
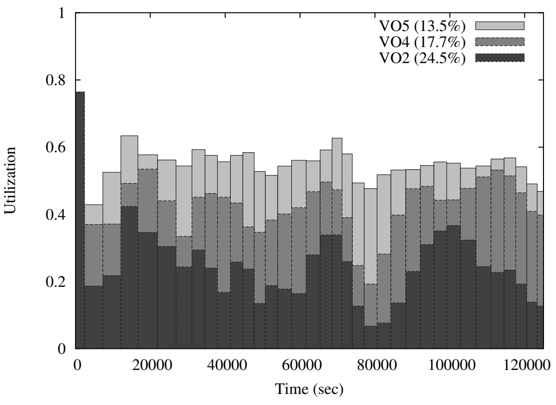


Figure 8. Utilizations of resource providers (with Bezier approximation)



(a) R1500



(b) R1000-1

Figure 9. Utilizations of VO users in resource providers R1500 and R1000-1

VOs, as well as their own VO resources. VO1 user shows the lowest acceptance rate due to lack of resources. Figure 7 shows the average unit cost for each VO user. The average cost is in inverse proportion to the job acceptance rate of Figure 6, since VO users with more resources can select the lower-cost resources.

Figure 8 shows the average utilization in each resource provider using Bezier approximation. The number in parenthesis indicates the average utilization during the simulations. Since R1500 and R1000-1 are used by various VO users throughout the hierarchy, their overall utilizations are high compared to other resources.

Figure 9 (a) and (b) show the utilization by each VO user in R1500 and R1000-1 respectively. As shown in Figure 9, resources in a VO are not only dynamically used by the VO users, but also by sub-VO users. In R1500, VO1 user occupies about 38% (=24.2%/64.0%) of the average utilization. VO2 and VO4 users share about 40% of the resource, while VO3 and VO5 users utilize the remaining 22%. In case of R1000-1, VO2 user occupies 44% of the resource utilization, while VO4 and VO5 users share the remaining 56% of the resource.

Next, we change the cost adjustment schemes of resource providers. The cost adjustment schemes in R1250, R1000-2, and R750 are kept as Dynamic-VO-Cost, while those in other resource providers are changed with Static-Cost and Dynamic-Load-Cost. Table 4 shows the average utilizations for three cost adjustment schemes of those three resource providers. In general, when Static-Cost is used, resource utilization becomes low compared to Dynamic-VO-Cost. Dynamic-Load-Cost shows more utilization since it adjusts the current cost according to the system load as well as the VO usage. When the system load is low, the low cost of the resource encourages users to use the resource. The utilization of R1500 with Static-Cost is high because users have no choice but to use R1500 when the costs of other resources are high.

Table 4. Average utilizations w.r.t. cost adjustment schemes

Resource provider	Average utilization (%)		
	Static-Cost	Dynamic-VO-Cost	Dynamic-Load-Cost
R1500	68.1	64.0	66.3
R1000-1	43.0	55.7	56.2
R500	31.6	39.7	40.1

6. Conclusions

In this paper, we define and model the hierarchical VO environment, in which a VO is composed of users, resource providers, and sub-VOs. Resource providers share their resources to multiple VOs including sub-VOs. Thus, users can access the resources in his or her own VO as well as in the ancestor VOs. In the proposed model, the resource allocation problem is formalized as mapping bags of tasks of a user's job using the user's accessible resource set in hierarchical VOs. We also provide a VO-wide resource allocation framework in resource brokers and suggest possible cost adjustment methods in resource providers. Simulation results show VO resources are used based on VO polices of hierarchical VOs.

Our future work includes the study on over-subscription problem in which the summation of resource shares to multiple VOs of a resource provider is more than 100%. Another issue is to analyze the effect of cost policy on the resource share. For example, the cost for local VO resources can be cheaper than other VOs in order to prioritize local users. We also plan to design and implement the proposed VO-wide resource framework.

References

[1] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: enabling scalable virtual organizations. *International Journal of Supercomputer Applications,* 15(3):200–222, 2001.

[2] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. In *Open Grid Service Infrastructure WG, Global Grid Forum*, 2002.

[3] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications and High Performance Computing,* 11(2):115–128, 1997.

[4] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing. In *Proceedings of 10th IEEE International Symposium on High-Performance Distributed Computing (HPDC 10),* San Francisco, USA, Aug. 2001.

[5] L. Pearlman, V. Welch, I. Foster, and C. Kesselman. A community authorization service for group collaboration. In *Proceedings of IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, pages 50–59, Monterey, USA, June 2002.

[6] Open Science Grid project. http://www.opensciencegrid.org.

[7] The Enabling Grids for E-sciencE project. http://www.eu-egee.org.

[8] J. In, P. Avery, R. Cavanaugh, and S. Ranka. Policy based scheduling for simple quality of service in Grid computing. In *Proceedings of the 18th Annual International Parallel and Distributed Processing Symposium (IPDPS),* Santa Fe, USA, April 2004.

[9] C. Dumitrescu and I. Foster. Usage policy-based cpu sharing in virtual organizations. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing (GRID 2004),* Pittsburgh, USA, November 2004.

[10] E. Elmroth and P. Gardfjall. Design and evaluation of a decentralized system for Grid-wide fairshare scheduling. In *Proceedings of 1st IEEE Conference on e-Science and Grid Computing*, Melbourne, Australia, December 2005.

[11] 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 Journal*, 18(8):1061–1074, 2002.

[12] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, A., Frohner, A. Gianoli, K. Lorentey, and F. Spataro. VOMS, an authorization system for virtual organizations. In *Proceedings of European Access Grids Conference*, pages 33–40, 2003.

[13] C. L. Dumitrescu and I. Foster. GRUBER: A Grid resource usage SLA broker. In *Proceedings of the 11th International European Parallel Computing Conference (EuroPar),* Lisbon, Portugal, 2005.

[14] A. Sulistio and R. Buyya. A time optimization algorithm for scheduling bag-of-task applications in auction-based proportional share systems. In *Proceedings of the 17th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD),* Rio de Janeiro, Brazil, October 2005.

[15] T. J. Norman, A. Preece, S. Chalmers, N. R. Jennigs, M. Luck, V. D. Dang, T. D. Nguyen, V. Deora, J. Shao, W. A. Gray, and N. J. Fiddian. Agent-based formation of virtual organizations. *Knowledge-Based Systems*, 17:103–111, 2004.

[16] W. Cirne, F. Brasileiro, J. Sauve, N. Andrade, D. Paranhos, E. Santos-Neto, and R. Medeiros. Grid computing for bag of tasks applications. In *Proceedings of the 3rd IFIP Conference on E-Commerce, E-Business and E-Government*, September 2003.

[17] R. Buyya and M. Murshed. GridSim: a toolkit for the modeling and simulation of distributed management and scheduling for Grid computing. *Concurrency and Computation: Practice and Experience,* 14(13):1175–1220, 2002.

[18] A. Sulistio, G. Poduvaly, R. Buyya, and C. K. Tham. Constructing a Grid simulation with differentiated network service using GridSim. In *Proceedings of the 6th International Conference on Internet Computing (ICOMP'05),* Las Vegas, USA, June 2005.