Fair Resource Sharing 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
111 Barry Street, Carlton, VIC, Australia
{jysh, raj}@csse.unimelb.edu.au

***Abstract*— In global Grid computing, 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 hierarchical VO environments. Resource providers and VOs agree upon VO resource sharing policies, such as resource sharing amount. Thus, users in lower-layer VOs can access resources in higher-layer VOs to accomplish their common goals. In this paper, we deal with fair resource allocation problem in hierarchical VOs, so that an appropriate proportion of a VO resource for each lower-layer VO is analyzed. In addition, we provide a resource allocation scheme based on these predefined proportions. Simulation results show that the proposed approach gives better fairness as well as performance compared with other schemes.**

I. Introduction

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]. 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 provide VO services and organize various VOs to utilize distributed resources efficiently [3], [4]. 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 has been conducted on policy-based resource allocation in VOs [5], [6], [7]. The resource broker allocates resources to a job according to the VO policies, such as the amount of resource share.

As the number of VOs increases in the Grid, efficient VO management is required. For example, Data Grids can be classified into four models in terms of organizations: monadic, hierarchical, federation, and hybrid [8]. Among various VO models, this paper focuses on the hierarchical VO model in which a VO can organize its own sub-VOs for the purpose of achieving the VO goal. Many national Grid systems have been established based on a consortium following the hierarchical VO model. Moreover, large-scale Grid application projects require hierarchical group structures for achieving their project goals.

In this paper, we deal with the resource allocation problem in the hierarchical VO-based Grids. Resource providers establish SLAs (Service Level Agreements) with their VOs that specify resource shares allowed to different VOs. Under such sharing polices of VOs, we provide internal sharing policies so that each VO user can be prioritized for efficient and fair use of resources. The main contributions of this paper are as follows: (i) to model hierarchical VO environments for global Grids based on resource sharing policy; (ii) to define the fair resource sharing problem in hierarchical VOs and provide a heuristic solution for it; and (iii) to propose and investigate resource allocation schemes.

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 in Section 3. Section 4 defines the fair resource sharing problem and provides a heuristic algorithm. In Section 5, we propose a resource allocation framework including the allocation scheme in the resource broker. We show simulation results in Section 6 and finally conclude the paper.

II. Related Work

Recent large-scale Grid projects include VO facilities to federate various distributed resources. The OSG (Open Science Grid) [3] provides a Grid infrastructure for large-scale scientific applications and enables resource sharing across VOs. The EGEE (Enabling Grids for E-SciencE) [4] also organizes many VOs and shares resources among them to increase efficiency. CAS (Community Authorization Service) [9] and VOMS (Virtual Organization Membership Service) [10] have been used to support authorization and authentication service for VOs.

Recent research on Grid computing has focused on polices for VO-wide scheduling and resource reservation. In [5], they introduce a new framework for policy based scheduling as a part of SPHINX scheduling system. The scheduling strategy in the framework adjusts resource usage accounts or request priorities for efficient resource usage management. Dumitrescu and Foster [6] 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 the resources in a site provided to a specific VO. They propose a prototype resource broker called GRUBER

[11] for resource usage SLA specification and enforcement in a Grid environment.

Elmroth and Gardfjall [7] have presented a decentralized architecture for a Grid-wide fair scheduling system, where each local scheduler enforces Grid-wide hierarchical sharing policies using global resource usage data. The policy engine calculates a fairshare priority factor for a job to support the Grid-wide share policy. Norman, et. al. [12] developed a model for 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 [12] is defined as a unit of economic services among users and service providers. Sulistio and Buyya [13] propose 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. In our previous work [14], we have formalized the resource allocation problem in hierarchical VOs and provided a cost optimization algorithm under different sharing policies. However, the proposed scheme did not show fairness in terms of resource distribution between different VO users when all the resource costs are the same.

Although the policy models in [5], [7] are based on a VO hierarchy, it is assumed that resource providers only define resource sharing of root VOs in VO policy trees, which is called local policy in [7]. All other VOs in policy trees follow the same share specified in the policy tree. In general, however, resource providers can negotiate with any VO in a VO hierarchy and provide different resource sharing polices. We investigate resource allocation in this general model.

Another approach in recent Grid research is resource co-allocation across multiple resource sites. In [15], they studied co-allocation in multicluster systems with both analytic means and with simulations for a wide range of parameters based on their previous work on influences of various parameters, such the job structure and size. Much research has also focused on scheduling and resource selection strategy of multi-site resources [16], [17]. The implementation of co-allocating scheduler are provided in [18], and a user-level scheduling Java API is also developed and introduced in [19]. In addition, a mechanism for advanced reservation for co-allocation of Grid resources is proposed in [20], while a negotiation model for supporting co-allocation is also examined in [21]. Although these co-allocation studies can utilize resources across multi-sites, they have not considered multiple VO environments.

III. Hierarchical Virtual Organizations

A. 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 [3], [14]. A VO can operate sub-VOs in order to accomplish the VO's goal.

For example, the left side in Fig. 1 shows a part of the consortium of APAC (Australian Partnership for Advanced Computing) Grid [22]. APAC consists of several partners, such as VPAC (Victorian Partnership for Advanced Computing), TPAC (Tasmanian Partnership for Advanced Computing), and AC3 (Australian Centre for Advanced Computing and Communications). VPAC is also a consortium of Victorian Member Universities, such as RMIT University and The University of Melbourne. On the other hand, the right side in Fig. 1 shows an example of BioGrid project, which is composed of Kidney Model, Drug Design, and Protein Folding groups. Thus, the hierarchical VO model in this paper can be applied to both physical and logical organizations.

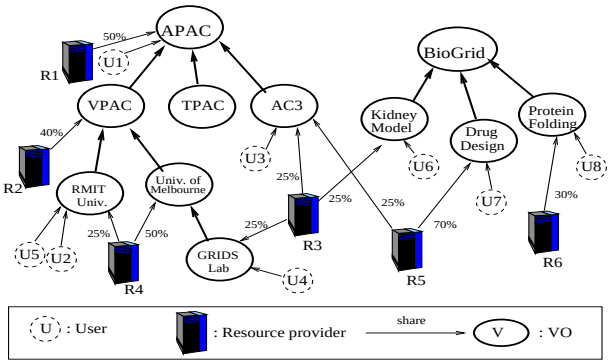


Fig. 1. An example of hierarchical VOs

As shown in Fig. 1, APAC consists of user U1, resource R1, and three sub-VOs (VPAC, TPAC, and AC3). BioGrid is composed of three sub-VOs. A sub-VO can include another sub-VOs, as in VPAC. Resource providers can share their resources to several VOs. For example, R3, R4, and R5 in Fig. 1 provide their resources to multiple VOs.

B. System Model

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* assigns different shares of resources to users in VOs that it has joined in. A *VO* is an organization of users, resource providers, and sub-VOs to meet the goal of that organization. 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. Table 1 shows the components of APAC VO in Fig. 1.

TABLE I
System components of APAC VO in Fig. 1

VO (v)	User (U_v)	Resource (R_v)	sub-VOs (V_v)
APAC	{U1}	{R1(50%)}	{VPAC, TPAC, AC3}
VPAC	∅	{R2(40%)}	{RMIT Univ., Univ. of Melbourne}
TPAC	∅	∅	∅
AC3	{U3}	{R3(25%), R5(25%)}	∅
RMIT Univ.	{U2, U5}	{R4(25%)}	∅
Univ. of Melbourne	∅	{R4(50%)}	{GRIDS Lab}
GRIDS Lab	{U4}	{R3(25%)}	∅

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). In hierarchical VOs, we additionally define the following terminologies.

- *Parent VO*: If a VO v is one of sub-VOs of 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 $ancs(v)$.
- *Descendent VOs*: All the VOs in the path from v to its leaf VOs in the VO tree are called *descendent* VOs of v. We denote it as $desc(v)$.
- *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 Fig. 1 as an example. Fig. 1 contains two root VOs (APAC and BioGrid), two intermediate VOs (VPAC and Univ. of Melbourne), and six leaf VOs.

2) Resource sharing policy model: In this paper, we consider VO polices between resource providers and their VOs in terms of resource sharing. The resource sharing policy of a resource provider r indicates the maximum amount of resource share to a VO v, which is denoted as $share(r, v)$. This sharing policy is a kind of SLA established between a resource provider and a VO. For example, R3 in Fig. 1 provides 25% of resource to AC3, 25% to GRIDS Lab, and 25% to Kidney Model VOs.

The resource share amount indicates the percentile of total resources in a resource provider. It has different meaning according to the resource provider's sharing policy. For the space-shared scheduling policy, the share amount implies the number of processors provided to VO. For the time-shared policy, it denotes the proportion in the total processing power of the resource provider assigned to VO. Our simulations in Section 6 use the time-shared scheduling policy.

3) Job model: A job in this paper is considered to be a bag-of-tasks application [23], which consists of multiple independent tasks with no communication among each other. In order to obtain the job's result, these tasks should be completed. In addition, we specify the deadline as a QoS parameter, so that the job execution must be finished before the deadline.

Thus, a user's job is defined as $(p, \{l_1, l_2, \cdots, l_p\}, d)$, 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.

IV. Fair Resource Sharing in Hierarchical VOs

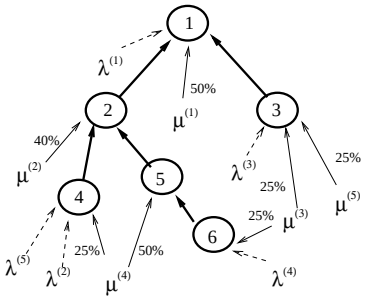
Since a VO can divide its operations into sub-VOs for achieving the VO goal, resource providers in a VO allow users in descendent VOs to use their resources as long as it does not violate the sharing policy. Thus, users can access resources in ancestor VOs as well as those in their own VOs. The resource broker should take this into consideration for resource allocation.

Suppose that resource capacity in a higher-level VO is better than the lower-level one. Then, users in lower-level are willing to use the higher-level VO resource due to its better performance. This can degrade QoS served by users in the higher-level VO and also lead to inefficient resource usage. Thus, we deal with the resource allocation problem in hierarchical VOs in order to provide fairness and efficiency.

Our approach consists of two parts: (i) determining internal resource sharing policy and (ii) allocating resource based on this policy. The internal resource sharing policy indicates the amount of resource which is prioritized to a specific VO's users. Thus, the resource allocation scheme selects these prioritized resources first. In the following subsections, we define the VO resource sharing problem and propose a heuristic algorithm to solve the problem. In Section 5, the resource allocation framework and scheme are provided.

A. Fair Resource Sharing Problem

Jobs are generated by users and arrive at VOs. We assume that user i submits jobs according to a Poisson process with the arrival rate $\lambda^{(i)}$. Each resource provider j is modeled as an M/M/1 queueing system with the average processing rate $\mu^{(j)}$. For example, hierarchical VOs of APAC in Fig. 1 are modeled as in Fig. 2.



(a) Job and service rates of VOs

VO	APAC	①
	VPAC	②
	AC3	③
	RMIT Univ.	④
	Univ. of Melbourne	⑤
	GRIDS Lab	⑥
Job arrival rate	Ui	$\lambda^{(i)}$
Service rate	Ri	$\mu^{(i)}$

(b) Notations

Fig. 2. VO model of APAC in Fig. 1

Now, we can derive a VO's job arrival rate from the participating users' job arrival rates. If two independent Poisson processes with the rates of λ_1 and λ_2 are merged, it follows a Poisson process with the rate of $\lambda_1 + \lambda_2$. Thus, a VO v's job arrivals are modeled as a Poisson process with the arrival rate λ_v in Eqn. (1).

$$\lambda_v = \sum_{u \in U_v} \lambda^{(u)} \tag{1}$$

Similarly, the resource processing rate of a VO v is defined by Eqn. (2). Only the shared amount of a resource provider is available to a VO, so that $\mu^{(r)}$ is multiplied by $share(r, v)$.

$$\mu_v = \sum_{r \in R_v} share(r, v) \times \mu^{(r)} \quad (2)$$

The problem of fair resource sharing considers how to allocate a VO's given processing rate to its descendent VOs for the purpose of minimizing the total waiting time. We denote the proportion of a VO i's service processing for a descendent VO j as $p_{i,j}$. Then, the service processing rate of the descendent VO j is increased by $p_{i,j} \cdot \mu_i$. The *actual service processing* rate of a VO i is defined by Eqn. (3).

$$\mu_i^a = \sum_{j \in \{i\} \cup ancs(i)} p_{j,i} \cdot \mu_j \quad (3)$$

Let us assume that the actual service processing rate and the job arrival rate of a VO i is known as μ_i^a and λ_i, respectively. The expected waiting time of VO i users is defined by $\frac{1}{\mu_i^a - \lambda_i}$. Thus, the problem *VO-SHARE* is:

minimize

$$\sum_{i \in V} \frac{1}{\mu_i^a - \lambda_i}$$

subject to

$$\sum_{j \in \{i\} \cup desc(i)} p_{i,j} = 1,$$

$$p_{i,j} \geq 0,$$

$$\mu_i^a > \lambda_i \quad \text{for all } i \in V.$$

B. Determining Resource Sharing

The *VO-SHARE* problem in the above is a *nonlinear optimization problem*. We provide a heuristic algorithm to solve the problem. First, we define a local optimization problem to allocate a VO i's resource to its descendent VOs. It is assumed that the actual service rate of each descendent VO j is known as μ_j^a. The problem is to decide $p_{i,i}$ and each $p_{i,j}$ where $j \in desc(i)$ to minimize the total waiting time ($\mu_i^a = 0$). Thus, the problem $SHARE_i$ is:

minimize

$$\sum_{j \in i \cup desc(i)} \frac{1}{p_{i,j} \cdot \mu_i + \mu_j^a - \lambda_j}$$

subject to

$$\sum_{j \in \{i\} \cup desc(i)} p_{i,j} = 1,$$

$$p_{i,j} \geq 0.$$

The proposed heuristics allocates the unit portion of the service processing rate of VO i to the resource provider j which minimizes the waiting time the most. Fig. 3 describes the pseudo-algorithm of determining local sharing. The output of the algorithm is $p_{i,j}$ of each participating VO, which is initially zero. The unit amount of allocated sharing is denoted as δ, so that the algorithm search the best VO which minimizes its waiting time the most for a given δ.

Algorithm Local_Fair_Share (i)
/* $w_{i,j}(p) = \frac{1}{p \cdot \mu_i + \mu_j^a - \lambda_j}$: waiting time function */
1: $p_{i,i} \leftarrow \frac{\lambda_i}{\mu_i} + \delta$;
2: **for** all j in $desc(i)$ **do** $p_{i,j} \leftarrow 0$;
3: $T \leftarrow \{i\} \cup desc(i)$;
4: **for** all j in T **do**
5: $\Delta w_{i,j} \leftarrow w_{i,j}(p_{i,j} + \delta) - w_{i,j}(p_{i,j})$;
6: $p \leftarrow p_{i,i}$;
7: **while** $p < 1.0$ **do**
8: $j \leftarrow \arg\max_{j \in T} \Delta w_{i,j}$.
9: $p_{i,j} \leftarrow p_{i,j} + \delta$;
10: $\Delta w_{i,j} \leftarrow w_{i,j}(p_{i,j} + \delta) - w_{i,j}(p_{i,j})$;
11: $p \leftarrow p + \delta$;
12: **endwhile**
13: **for** all j in T **do**
14: $\mu_j^a \leftarrow \mu_j^a + p_{i,j} \cdot \mu_i$;

Fig. 3. Heuristic for local optimization

The initial value of $p_{i,i}$ is set with $\frac{\lambda_i}{\mu_i} + \delta$ because it is the minimum condition for an M/M/1 queue to be stable. The reduced amount of the waiting time of VO j by allocating δ is denoted as $\Delta w_{i,j}$ (line 5). Thus, the algorithm selects the VO of which $\Delta w_{i,j}$ is the largest (line 8). At the end of the algorithm, $p_{i,j}$ is determined and updated to μ_j^a (line 14).

The *VO-SHARE* problem is solved based on $SHARE_i$. Fig. 4 describes the pseudo-algorithm of the fair resource sharing. The algorithm **Fair_Share** (v) recursively decides the proportion of resource sharing. It first determines all child VOs' shares (line 7-8). If a VO is a leaf in the hierarchy, it

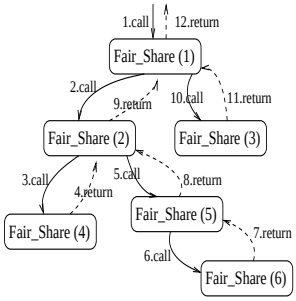
Algorithm VO_Share (G)
/* - $G = (U, R, V)$: a Grid with VOs */
1: **for** each VO v in V **do**
2: $\lambda_v = \sum_{u \in U_v} \lambda^{(u)}$
3: $\mu_v = \sum_{r \in R_v} share(r, v) \times \mu^{(r)}$
4: **endfor**
5: **for** each root VO $r \in V$ **do**
6: **Fair_Share** (r);

Algorithm Fair_Share (v)
7: **for** all child VO i in V_v **do**
8: **Fair_Share** (i);
9: **if** $V_v == \phi$ **then** /* leaf VO */
10: $p_{v,v} \leftarrow 1.0$;
11: $\mu_v^a \leftarrow \mu_v$;
12: **else** /* non-leaf VO */
13: **Local_Fair_Share** (v);
14: **endif**

Fig. 4. Fair VO resource sharing

allocates all the resource to itself (line 9-11). Otherwise, a VO allocates its resource to its descendent VOs by the algorithm *Local_Fair_Share ()* as shown in Fig. 3. Since the Grid system consists of multiple hierarchical VOs, the fair resource sharing can be obtained by investigating root VOs in the system (line 5-6).

For example, Fig. 5(a) shows the sequence of invoked function calls and returns of *Fair_Share(1)* in Fig. 2. As shown in Fig. 5(a), the proportion of resource sharing of a VO is determined after all values of its sub-VOs are determined. The first column of Fig. 5(b) indicates the depth-first traversal of function calls. As a result, the second column of Fig. 5(b) shows determined value of $p_{i,j}$ at each function return point.



(a) Function call and return sequence

Return sequence	Determined value
4. return Fair_Share (4)	$p_{4,4}$
7. return Fair_Share (6)	$p_{6,6}$
8. return Fair_Share (5)	$p_{5,5}$ $p_{5,6}$
9. return Fair_Share (2)	$p_{2,2}$ $p_{2,4}$ $p_{2,5}$ $p_{2,6}$
11. return Fair_Share (3)	$p_{3,3}$
12. return Fair_Share (1)	$p_{1,1}$ $p_{1,2}$ $p_{1,3}$ $p_{1,4}$ $p_{1,5}$ $p_{1,6}$

(b) Determined values at function return

Fig. 5. Determining $p_{i,j}$ of Fig. 2

V. VO-wide Resource Allocation

A. 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. Fig. 6 shows the system components of hierarchical VOs in Fig. 2. VO resource brokers (VO-RBs) cooperate with their parent and sub-VOs as shown in Fig. 6. The followings are 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 along with the 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 ancestor VOs in the VO policy tree. The user can access the resources of the ancestor VOs because the user's job is run not only for the VO itself but also for the ancestor VOs.

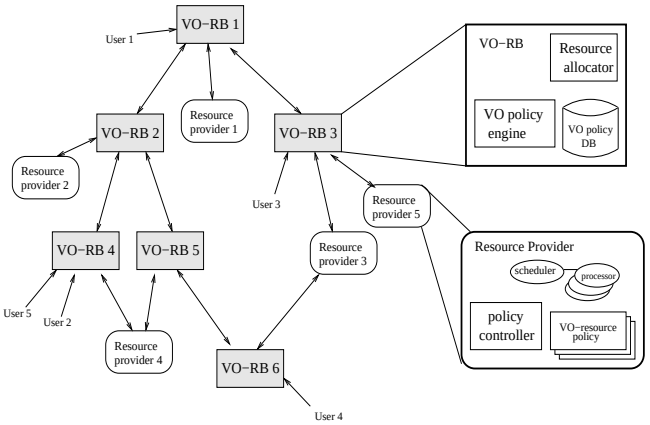


Fig. 6. The VO-wide resource allocation framework

(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 handled by the local scheduler in each resource provider.

(4) *Updating sharing polices.* If a 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.

Each resource provider has the local scheduler which accepts submitted tasks and schedules those tasks based on its own scheduling policy. The local scheduler accepts only tasks that can meet their QoS requirements. The policy controller in the resource provider contacts the resource broker and informs changed policy and status, such as the current system load or sharing policy.

Besides job allocation, the resource broker monitors the job arrival rates of users and periodically updates the job arrival rate, which results in changing the proportion of resource usage prioritized to VOs. In such case, the resource broker informs its root VO in order to update the resource sharing. Then, the root VO resource broker initiates the fair sharing algorithm of Fig. 4 and enforces the changed policy to all descendent VO resource brokers. This process of policy update also happens when a resource provider joins or leaves a VO, or changes the sharing policy.

B. Resource Allocation Scheme

The VO resource broker manages several VO polices and data structures for resource allocation. The followings are such polices in the resource broker of VO v.

- $share_v^{max}$: The maximum amount of resources shared by resource providers in the VO. It is obtained by adding all sharing resources from resource providers of a VO.

Algorithm VO_wide_Resource_Allocation (J, v)

/* - $J = (p, \{l_1, \cdots, l_p\}, d)$: a job
- v : a VO
*/

1: $task\_index \leftarrow 1$;
2: $i \leftarrow v$;
3: **while** $i \neq \phi$ **do**
4: **while** $u_{i,v} < p_{i,v}$ **do**
5: **for** each $r \in R_i$ **do**
6: $(alloc, load) \leftarrow$ *Submit* $(J, task\_index, r, i)$;
7: $task\_index \leftarrow task\_index + alloc$;
8: $u_{i,v} \leftarrow u_{i,v} + load$;
9: **if** $task\_index > p$ **then return accept**;
10: **endfor**
11: **endwhile**
12: $i \leftarrow parent(i)$;
13: **endwhile**
14: $i \leftarrow v$;
15: **while** $i \neq \phi$ **do**
16: **while** $share_i^{curr} < share_i^{max}$ **do**
17: **for** each $r \in R_i$ **do**
18: $(alloc, load) \leftarrow$ *Submit* $(J, task\_index, r, i)$;
19: $task\_index \leftarrow task\_index + alloc$;
20: $share_i^{curr} \leftarrow share_i^{curr} + load$;
21: **if** $task\_index > p$ **then return accept**;
22: **endfor**
23: **endwhile**
24: $i \leftarrow parent(i)$;
25: **endwhile**
26: Cancel all allocated sub-tasks in the above.
27: **return reject**;

Fig. 7. Resource allocation scheme

- $share_v^{curr}$: The current amount of resource used in the VO. It is defined by the required resource of currently accepted jobs divided by the total amount of resource.
- $p_{v,v'}$: The proportion of resource usage prioritized to descendent VO v'. It is derived as in Section 4.
- $u_{v,v'}$: The current amount of resource used in a descendent VO v'.

The VO resource broker aims to meet the job deadline as a QoS requirement under VO resource policies. Fig. 7 shows the pseudo resource allocation algorithm of the VO resource broker.

The allocation scheme selects the VO's resource providers first, and then traverses resources in other ancestor VOs. The function **Submit** in line 6 of Fig. 7 sends the job along with information of the task index to schedule ($task\_index$) to the selected resource r. The local scheduler of the resource r accepts only sub-tasks that can meet their deadlines and returns the number of allocated tasks ($alloc$) and resource usage ($load$). The policy controller of a resource provider enforces the sharing policy so that the total resource usage in r cannot exceed $share(r, v)$ in a VO v.

The resource allocation consists of two steps: internal policy-based and external policy-based allocations. In the first while-loop in Fig. 7 (line 3 $\sim$ 13), the resource usage policy follows the internal sharing policy ($p_{i,j}$). Each VO has the resource proportion which is prioritized to the VO. Thus, the resource broker first allocates resource under the internal sharing policy which corresponds to the while-loop condition in line 4 in Fig. 7. The remaining sub-tasks after the internal policy-based allocation are allocated with resources under the external policy ($share(r, v)$). The second while-loop from line 15 to line 25 corresponds to the second resource allocation.

If all p sub-tasks are successfully allocated, the algorithm ends and the job is accepted (line 9 and line 21). However, if there is no sufficient resources to run the job, it cancels all the previously allocated sub-tasks and rejects the job (line 26 $\sim$ 27). The user can submit the rejected job again later, or the resource broker can manage the waiting queue for those rejected jobs.

VI. Simulation Results

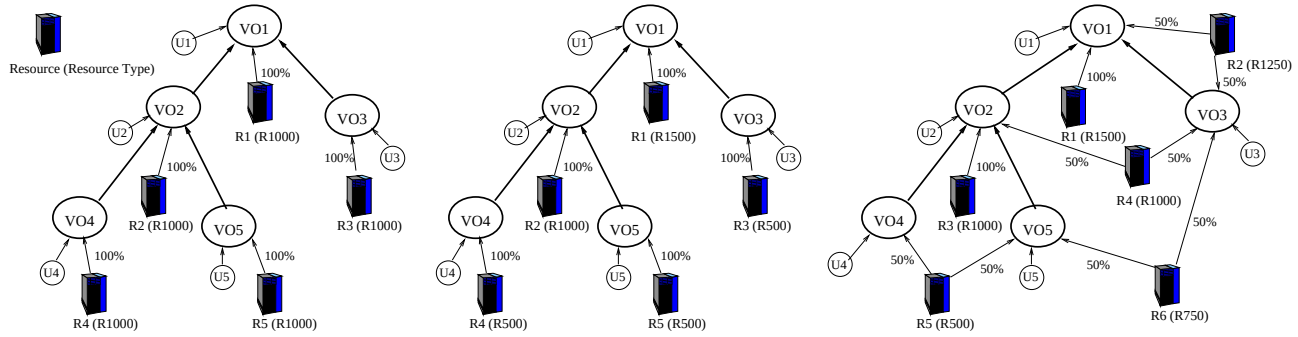
In this section, we simulate the proposed resource allocation scheme using the GridSim toolkit [24], [25]. Fig. 8 shows the simulated hierarchical VO environments with three different scenarios of resource providers. We use five resource types as shown in Table II. In scenario 1 and 2 (Fig. 8(a) and (b)), each VO has one dedicated resource provider. All resource providers have the same resource capacity in scenario 1. Resource providers in scenario 2 are assumed to provide different capacity (R1 $>$ R2 $>$ R3 = R4 = R5). In scenario 3 (Fig. 8(c)), R1 and R3 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.

We compare the proposed scheme with other resource allocation schemes including Least Load First (LLF), Random, and Round Robin, as in [6]. The LLF scheme selects the resource provider with the lowest current load among possible resources to access in hierarchical VOs. Random and Round

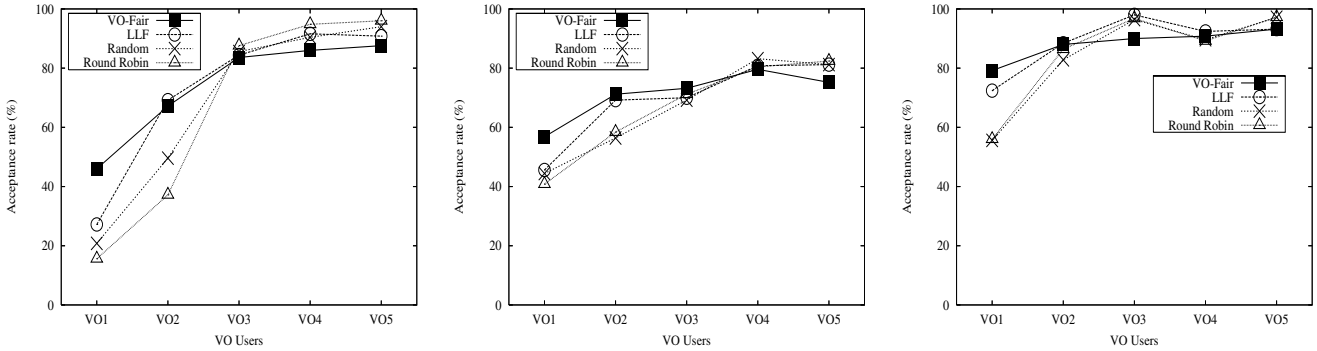
TABLE II
Resource Characteristics

Resource type	Processor performance (MIPS)	Number of processors
R1500	1500	20
R1250	1250	20
R1000	1000	20
R750	750	20
R500	500	20



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Fig. 8. Simulated VO environments



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Fig. 9. Average acceptance rates of VO users in simulations

Robin schemes select a resource provider among accessible resources randomly and in round robin, respectively. The proposed resource allocation scheme is denoted as **VO-Fair**. We fix δ as 0.01 in simulations shown in Fig. 8.

If a user's job can meet the deadline, it is accepted and allocated to a resource provider. We use the average acceptance rate of a user's job submission as a metric. Fig. 9 shows the average acceptance rates of VO users in the simulations. In scenario 1 (Fig. 8(a)), each resource provider has the same resource capacity. The acceptance rates of VO1 and VO2 are lower than others because other VO users share their resources. On the contrary, VO3, VO4, and VO5 users show higher acceptance rates since they can access resource providers in higher VOs. VO1 and VO2 of VO-Fair show higher acceptance rates compared to other schemes.

In scenario 2 results (Fig. 9(b)), VO1 and VO2 users of other schemes still show lower acceptance rates, although resource providers in VO1 and VO2 have better performance. However, the proposed VO-Fair shows better fairness because all VO users' jobs are accepted similarly. In scenario 3, we simulate more complicated resource sharing policy as shown in Fig. 8(c). Since VO3 in Fig. 8(c) are provided with many resource providers, VO3 user shows the highest job acceptance rate in other schemes except the proposed one. However, users in the proposed scheme show similar acceptance rates, as shown in Fig. 9(c).

Table III shows average acceptance rates and standard deviations of Fig. 9. The proposed scheme shows higher acceptance rates in all scenarios. Moreover, lower standard deviations in Table III indicate that the proposed scheme provides better fairness.

TABLE III
SIMULATION RESULTS

	Strategy	Scenario 1	Scenario 2	Scenario 3
Average acceptance rate(%)	VO-Fair	74.08	71.2	88.88
	LLF	72.64	69.36	88.24
	Random	68.08	66.88	84.32
	Round Robin	66.24	66.64	85.12
Standard deviation	VO-Fair	17.69	8.63	5.39
	LLF	26.95	14.46	9.82
	Random	31.84	16.52	17.08
	Round Robin	37.30	17.28	16.95

VII. CONCLUSIONS

In this paper, we proposed a resource allocation scheme based on fair resource sharing in hierarchical VOs. We derived the internal fair sharing policies under the given sharing resource policies of hierarchical VOs. VO resource brokers

manage their VO policies and member status so that they co-operate with each other to provide efficient resource allocation. Simulation results show that the proposed scheme provides greater fairness than other schemes, as well as better performance.

Based on the proposed framework, we are currently implementing VO-based resource brokering in Gridbus broker [26]. We will investigate the practical issue throughout this implementation, such as scalability and broker system overhead. Our future work also includes the study of the over-subscription problem, in which the summation of resource shares assigned to multiple VOs of a resource provider is more than 100%.

ACKNOWLEDGEMENTS

This work is partially supported by ARC Discovery Project and DEST International Science Program grants. In addition, we would like to thank Srikumar Venugopal, James Andrew Broberg, and the anonymous reviewers for their valuable comments.

REFERENCES

[1] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the Grid: Enabling scalable virtual organizations," *International Journal of Supercomputer Applications*, vol. 15, no. 3, pp. 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] Open Science Grid project. http://www.opensciencegrid.org

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

[5] 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 2004)*, Santa Fe, USA, Apr. 2004.

[6] 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, Nov. 2004, pp. 53–60.

[7] 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, Dec. 2005.

[8] S. Venugopal, R. Buyya, and K. Ramamohanarao, "A taxonomy of data grids for distributed data sharing, management and processing," *ACM Computing Surveys*, vol. 38, no. 1, pp. 1–53, 2007.

[9] 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*, Monterey, USA, June 2002, pp. 50–59.

[10] 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*, 2003, pp. 33–40.

[11] 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.

[12] 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*, vol. 17, pp. 103–111, 2004.

[13] 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*, Rio de Janeiro, Brazil, Oct. 2005.

[14] K. H. Kim and R. Buyya, "Policy-based resource allocation in hierarchical virtual organizations for global grids," in *Proceedings of the 18th International Symposium on Computer Architecture and High Performance Computing*, Ouro Preto, Brazil, Oct. 2006.

[15] A. I. D. Bucur and D. H. J. Epema, "The maximal utilization of processor co-allocation in multicluster systems," in *Proceedings of International Parallel and Distributed Processing Symposium (IPDPS 2003)*, Nice, France, Apr. 2003, pp. 60–69.

[16] C. Ernemann, V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour, "Enhanced algorithms for multi-site scheduling," in *Proceedings of the 3rd International Workshop on Grid Computing*, Baltimore, USA, Nov. 2002, pp. 219–231.

[17] W. Zhang, A. M. K. Cheng, and M. Hu, "Multisite co-allocation algorithms for computational Grid," in *Proceedings of 20th International Parallel and Distributed Processing Symposium (IPDPS 2006)*, Rhodes Island, Greece, Apr. 2006.

[18] H. H. Mohamed and D. H. J. Epema, "Experiences with the KOALA co-allocating scheduler in multiclusters," in *Proceedings of International Symposium on Cluster Computing and the Grid*, Cardiff, UK, May 2005, pp. 784–791.

[19] H. Nakada, A. Takefusa, K. Ookubo, M. Kishimoto, T. Kudoh, Y. Tanaka, and S. Sekiguchi, "Design and implementation of a local scheduling system with advance reservation for co-allocation on the Grid," in *Proceedings of 6th IEEE International Conference on Computer and Information Technology*, Seoul, Korea, Sept. 2006.

[20] M. Siddiqui, A. Villazon, R. Prodan, and T. Fahringer, "Advance reservation and co-allocation of Grid resources: A step towards an invisible Grid," in *Proceedings of 9th IEEE International Multitopic Conference*, Karachi, Pakistan, Dec. 2005, pp. 1–6.

[21] J. Li and R. Yahyapour, "Negotiation model supporting co-allocation for Grid scheduling," in *Proceedings of 7th IEEE/ACM International Conference on Grid Computing*, Barcelona, Spain, Sept. 2006, pp. 254–261.

[22] APAC: National Grid for Australian eResearch. http://www.grid.apac.edu.au

[23] 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*, Sept. 2003.

[24] 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*, vol. 14, no. 13-15, pp. 1175–1220, 2002.

[25] A. Sulistio, G. Poduval, R. Buyya, and C. K. Tham, "On incorporating differentiated levels of network service into GridSim," *Future Generation Computer Systems (FGCS)*, vol. 23, no. 4, pp. 606–615, 2007.

[26] S. Venugopal, R. Buyya, and L. Winton, "A Grid service broker for scheduling e-Science applications on global data Grids," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 6, pp. 685–699, 2006.