Contention-aware Resource Management System in a Virtualized Grid Federation

Mohsen Amini Salehi, Rajkumar Buyya
Cloud Computing and Distributed Systems (CLOUDS) Laboratory
Department of Computer Science and Software Engineering The University of Melbourne, Australia
{mohsena,raj}@csse.unimelb.edu.au

***Abstract*—Several efforts have been made towards the development of architectures, mechanisms, and policies that enable resource allocation across several Grids. Recently, such federated Grids have applied lease-based and Virtual Machine-based approaches in allocating resources. Lease-based approach along with VMs impose new challenges to the resource management systems in federated Grids. In this research we deal with some of these challenges in the context of InterGrid as a virtualized federated Grid environment. The first challenge we deal with is selecting the best set of leases for preempting in favor of users with higher priority. The second problem we consider in federated Grid is how a meta-scheduler should schedule the incoming requests amongst available sites (clusters) in a way that minimum contention take place.**

I. Introduction

Federated Grids enable sharing, selection, and aggregation of resources across several Grids, which are connected through high bandwidth network connections. Resource provisioning for user applications is one of the main challenges and research areas in federated Grid environments. Job abstraction is widely used in resource management of Grid environments. However, due to advantages of Virtual Machine (VM) technology, recently, many resource management systems have emerged to enable another style of resource management based on the *lease* abstraction [6].

InterGrid, as a federated Grid environment, also aims to provide a software system that interconnects islands of virtualized Grids. It provides resources in the form of VMs and allows users to create execution environments for their applications on the VMs [5]. In each constituent Grid, the provisioning rights over several clusters inside the Grid are delegated to the InterGrid Gateway (IGG). On the other hand, local users in each cluster send their requests directly to the local resource manager (LRM) of the cluster.

Hence, resource provisioning is done for two different types of users, namely: local users and external users. As illustrated in Figure 1, local users (hereafter termed as local requests), refer to users who ask their local cluster resource manager (LRM) for resources. External users (hereafter termed as external requests) are those users who send their requests to a gateway (IGG) to get access to larger amount of shared resources. Typically, local requests have priority over external requests in each cluster [4]. In other words, the organization that owns the resources would like to ensure that its community has priority access to the resources. Under such a circumstance, external requests are welcome to use resources if they are available. Nonetheless, external requests should not delay the execution of local requests. The first problem we deal with in this environment is how to

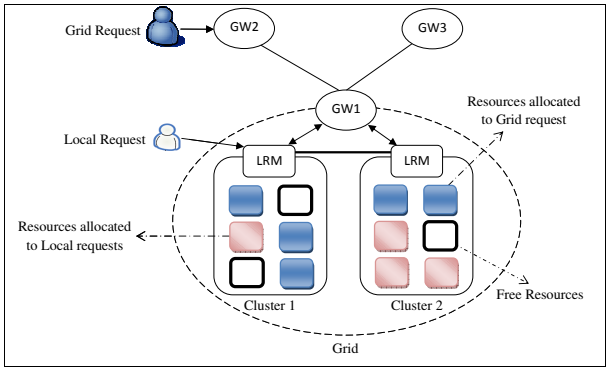


Fig. 1. A scenario that shows the contention between local and external requests in a resource sharing environment

remove the contention between local and external requests in InterGrid. Given the fact that local requests have more priority rather than local requests we preempt external leases in favor of local requests. More specifically, we propose policies to determine an appropriate set of leases for preemption. Additionally, we make an appropriate decision for the preempted lease based on the VMs capabilities.

The second problem that we address in this research is how the contention between local and external requests that take place in a virtualized Grid environment can be decreased. More specifically, we look at this problem from a meta-scheduling point of view, where the meta-scheduler distributes incoming requests (from other Grids in the case of InterGrid) between several clusters within that Grid.

The rest of this paper is organized as follows: In Section II, an overview of the InterGrid environment is provided. Section III introduces preemption policy. Next, in Section IV meta-scheduling solution for reducing contention is provided. Finally, conclusion and future works are provided in Section V.

II. Background and Context

InterGrid aims to provide a framework that allows users to create execution environments for various applications on top of the physical infrastructure participating in Grid systems. Peering arrangements established between gateways

(termed IGG in the InterGrid context) enable the allocation of resources from different Grids to fulfill the requirements of the execution environments.

The Local Resource Manager (LRM)[1] is the resource manager in each cluster and provisions resources for the local and external requests. Virtual Machine (VM) technology is used in each cluster of InterGrid for resource provisioning.

In the InterGrid each request is contiguous and must be served within resources of a single cluster. Each request has a type, number of VMs, duration, and the deadline (optional). We consider several types of Grid requests in InterGrid. These Grid requests can broadly be classified as Best-Effort (BE) and Deadline-Constraint (DC) requests. BE Grid requests can be preempted in favor of local requests. If there is not enough resources to start BE requests, they are scheduled in the first available time-slot. DC Grid requests cannot be preempted if the deadline is tight. Additionally, DC requests are rejected if there is not enough resources for them to start.

BE Grid requests can be either *Cancelable:* which can be started at any time and is terminated in the case of preemption; or *Suspendable:* which can be started at any time and is rescheduled in later time-slot in the case of preemption. DC Grid requests can be *Migratable:* which are sent to another cluster inside the same Grid in the case of preemption; or *Non-preemptive:* which cannot be preempted at all. We also consider local requests of a cluster as Non-preemptive requests. To see more details about different Grid request types readers can refer to [2], [5].

III. Preemption Policy

When a local request cannot be allocated because resources are occupied by external requests, preemption occurs to free space for local requests. Specifically, allocation of parallel requests can potentially result in different leases to be preempted. Each *candidate set* contains a set of leases that their preemption makes enough space for an incoming local request. From the system centric perspective, choosing different candidate sets affects the number of VMs to be preempted, which turns out to present different time overheads and different resource utilization. From user centric perspective, selecting different candidate sets leads to a different number of leases to be preempted, which adds more waiting time and, consequently, more external user dissatisfaction.

We propose an approach (MOML) that can fulfill both system and user centric criteria at the same time. This policy is shown in Algorithm 1.

According to Algorithm 1, in MOML the selection of a candidate set is carried out in two phases. In the first phase (pre-selection) all candidate sets which have a total overhead less than a certain threshold (α) are pre-selected for the second phase (lines 5 to 8 in Algorithm 1). In the second phase, to have fewer leases affected, a candidate set that contains minimum number of leases is selected(lines 9 to 11 in Algorithm 1). To keep the trade-off between overhead and waiting time, we consider α as the *median* value of the overheads (lines 1, 2, and 4 in Algorithm 1). By choosing $\alpha = median$ we ensure that just half of the candidate sets that have lower overheads are considered in the second phase for having a minimum number of leases. In the experiments we compare the resource utilization and number of lease preemption resulted from different policies. We compare MOML against 2 other policies. MOV: which minimizes preemption overhead time and MLIP: which minimizes the number of leases preempted.

Algorithm 1: MOML Preemption Policy.

```
   Input: Candidate Sets
   Output: Selected Candidate Set
1  foreach candidateSet ∈ Candidate Sets do
2      Overheads.Add(getOverhead(candidateSet));
3  min ← ∞;
4  α ← getMedian(Overheads);
5  foreach candidateSet ∈ Candidate Sets do
6      ovhd ← getOverhead(candidateSet);
7      NoLeases ← Cardinality(candidateSet);
8      if ovhd ≤ α then
9          if NoLeases < min then
10             selected ← candidateSet;
11             min ← NoLeases;
```

Figure 2, shows the impact of altering workload parameters on resource utilization. This experiment indicates that resource utilization increases almost linearly by increasing the average number of VMs in requests (Figure 2(a)). Although preempting best effort leases make some overhead, we can see in Figure 2(b) that increasing the number of best effort requests improves resource utilization.

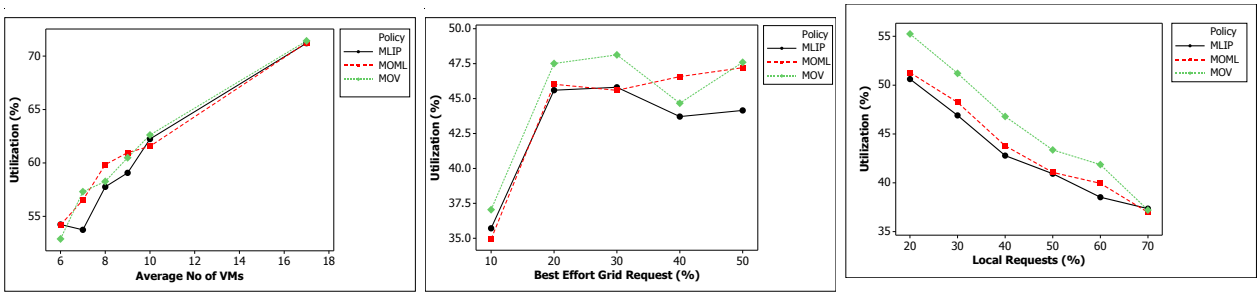
By increasing the number of local requests the number of preemption and subsequently the amount of overhead increases. Therefore, as we can see in Figure 2(c), by increasing the number of local requests, resource utilization decreases almost linearly in all policies.

Figure 3(a) shows that, in general, leases with more number of VMs lead to fewer of preemptions. In fact, in this situation fewer of leases are needed to be preempted to make room for incoming local requests.

Figure 3(b) shows that by increasing the number of best effort external requests the number of preemptions increases almost linearly. For a lower percentage of best effort external requests (percentage best effort<30%), MOML behaves similar to MOV, however, after that point MOML approaches to MLIP.

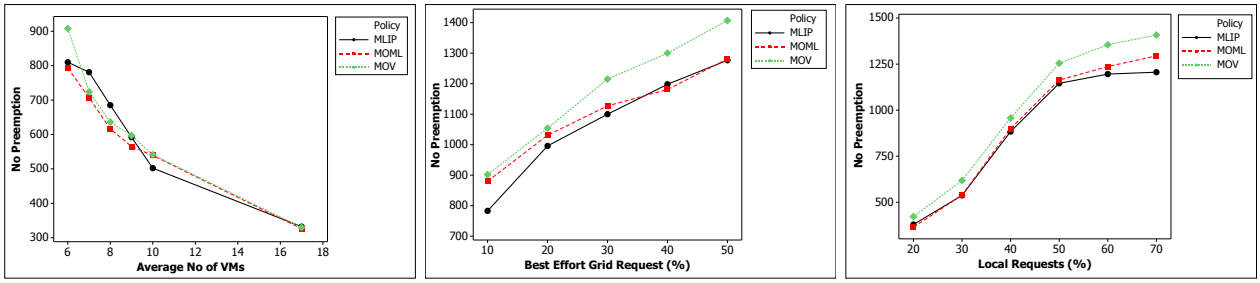
Figure 3(c) demonstrates that the number of preemptions increases by increasing the number of local requests in all

[1]This component is also called Virtual Infrastructure Engine (VIE) in the InterGrid.



(a) (b) (c)

Fig. 2. Resource utilization by modifying (a) average number of VMs, (b) the percentage of best effort external requests, and (d) the percentage of local requests.



(a) (b) (c)

Fig. 3. Number of lease preemption by changing (a) the average number of VMs, (b) percentage of best effort external requests, (c) the number of local requests.

policies almost linearly. As illustrated in all sub-figures of Figure 3, most of the time MLIP results in a minimum number of preemptions and MOML operates between MLIP and MOV.

IV. Contention-aware Meta-scheduling

In this section we propose scheduling policy in the meta-scheduler of InterGrid (IGG) that determines the fraction of external requests that should be allocated to each cluster in a way that minimizes the contention (number of VM preemptions). The proposed policy is based on the stochastic analysis of routing in parallel, non-observable queues. To see details of the analysis interested readers can refer to [1]. However, due to space shortage, here, we just describe the 2 policies that we proposed namely: *workload allocation policy* and *dispatch policy*. Workload allocation policy determines the proportion of external requests that have to be sent to each cluster in a Grid. Dispatch policy determines the sequence of submitting different types of external requests to different clusters in a way that external requests with tighter QoS affect less by preemption.

The preemption-aware workload allocation policy (PAP) is presented in the form of pseudo-code in Algorithm 2. According to Algorithm 2, at first ψ is calculated for all clusters. Then, in steps 4 to 10, to exclude the heavily loaded clusters, clusters are sorted based on the ψ value in the ascending order. Next, the value of k is increased up until condition defined in step 7 is met. ub is found by starting from $2 \cdot lb$ and is doubled up until condition in step 13 is met. Steps 16-21 show a bisection algorithm for finding proper value for z. Finally, in steps 22 and 23 the arrival rate to each cluster is determined. Steps 24 and 25 guarantee that clusters $k+1$ to N, which are heavily loaded, do not receive any external request.

The algorithm proposed above determines the routing probability to each cluster. However, it does not offer any deterministic sequence for dispatching each external request to the clusters. More importantly, as mentioned earlier, external requests are in different levels of QoS which implies that some external requests are more valuable. Hence, we would like to decrease the chance of preemption for more valuable requests to the minimum possible. We put this precedence in place through a dispatch policy.

In this part, we propose a policy that, firstly, reduces the number of VM preemptions for more valuable external requests; Secondly, this policy makes a deterministic sequence for dispatching external requests. It is worth noting that the dispatch policy uses the same routing probabilities that worked out for each cluster using workload allocation policy. The only difference is in the sequence of requests dispatched to each cluster. For this purpose, we adapt Billiard strategy [3] as the dispatching policy.

Minimizing the likelihood of preempting valuable requests depends on the scheduling policy in the gateway (IGG) (which is investigated in this paper) as well as the local scheduling policy in each cluster. The local scheduling policy we use in the clusters has the awareness of the request types and at the time of preemption preempts leases that belong to less valuable users [2]. Given this policy for the local schedulers of each cluster, more valuable leases would be preempted if and only if there is not (sufficient) leases of less valuable request types to be preempted. We can infer that the

Algorithm 2: Preemption-aware workload allocation Policy (PAP).

Input: $\bar{\Lambda}_j$,θ_j,ω_j,λ_j,τ_j,μ_j, for all $1 \leq j \leq N$.
Output: ($\hat{\Lambda}_j$) load distribution of external requests to different clusters, for all $1 \leq j \leq N$.

1 **for** $j \leftarrow 1$ **to** N **do**
2 $\quad \psi_j = \frac{\lambda_j \mu_j}{2(1-\rho_j)^2} + \frac{\theta_j}{(1-\rho_j)}$;
3 //Sort array ψ in ascending order;
4 Sort (ψ);
5 $k \leftarrow 1$;
6 **while** $k < N$ **do**
7 $\quad$ **if** $\sum_{j=1}^{k} \phi_j(\psi_k) \geq \left(\sum_{j=1}^{k} \frac{(1-\rho_j)}{\theta_j} \right) - \Lambda$ **then**
8 $\quad\quad$ break;
9 $\quad$ **else**
10 $\quad\quad$ $k \leftarrow k + 1$;
11 $lb \leftarrow \psi_k$;
12 $ub = 2 * lb$;
13 **while** $\sum_{j=1}^{k} \phi_j(ub) > \left(\sum_{j=1}^{k} \frac{(1-\rho_j)}{\theta_j} \right) - \Lambda$ **do**
14 $\quad$ $ub = 2 * ub$;
15 //ϵ is the expected precision;
16 **while** $ub - lb > \epsilon$ **do**
17 $\quad$ $z \leftarrow (lb + ub)/2$;
18 $\quad$ **if** $\sum_{j=1}^{k} \phi_j(z) \geq \left(\sum_{j=1}^{k} \frac{(1-\rho_j)}{\theta_j} \right) - \Lambda$ **then**
19 $\quad\quad$ $lb \leftarrow z$;
20 $\quad$ **else**
21 $\quad\quad$ $ub \leftarrow z$;
22 **for** $j \leftarrow 1$ **to** k **do**
23 $\quad$ $\hat{\Lambda}_j = \frac{(1-\rho_j)}{\theta_j} - \frac{1}{\theta_j} \sqrt{\frac{(1-\rho_j)(\omega_j(1-\rho_j))+\theta_j \lambda_j \mu_j}{2\theta_j(1-\rho_j)z+(\omega_j - 2\theta_j^2)}}$;
24 **for** $j \leftarrow k+1$ **to** N **do**
25 $\quad$ $\hat{\Lambda}_j = 0$;

chance of getting preempted for valuable external requests would be low if a *mixture* of valuable and less valuable external requests are dispatched to each cluster. Therefore, in the dispatch policy we keep track of number of external requests of each type that are dispatched to each cluster.

The pseudo-code developed for this purpose is presented in Algorithm 3. In Algorithm 3, at first the fastest cluster is found based on the average service time for external requests in each cluster (step 1). We consider P_j^i as the probability of dispatching request type i to cluster j. P_j^i is worked out based on P_j and the proportion of request type i in external requests (steps 4, 5). In step 7, we assign 1 to the fastest cluster. Y_j^i expresses the number of external requests of type i that are dispatched to cluster j and initially is zero (step 6). By receiving an external request, value of the adapted billiard sequence for all clusters are worked out and a cluster with

Algorithm 3: Request Type Dispatch Policy (RTDP).

Input: P_j,θ_j for all $1 \leq j \leq N$.
Output: $SelectedCluster(j_s)$

1 $fastestCluster \leftarrow$ findFastestCluster(θ);
2 **foreach** *Cluster* j **do**
3 $\quad$ $X_j \leftarrow 0$;
4 $\quad$ **foreach** *RequestType* i **do**
5 $\quad\quad$ $P_j^i \leftarrow P_j *$ GetProportion(i);
6 $\quad\quad$ $Y_j^i \leftarrow 0$;
7 $X_{fastestCluster} \leftarrow 1$;
8 **foreach** *external request received* **do**
9 $\quad$ $i \leftarrow$ GetRequestType();
10 $\quad$ $min \leftarrow MaxValue$;
11 $\quad$ **foreach** *Cluster* j **do**
12 $\quad\quad$ **if** *($P_j^i \neq 0$)* **then**
13 $\quad\quad\quad$ $D = (X_j + Y_j^i)/P_j^i$;
14 $\quad\quad\quad$ **if** *($D < min$)* **then**
15 $\quad\quad\quad\quad$ $min \leftarrow D$;
16 $\quad\quad\quad\quad$ $tmpCluster \leftarrow j$;
17 $\quad$ $Y_{tmpCluster}^i \leftarrow Y_{tmpCluster}^i + 1$;
18 $\quad$ $j_s \leftarrow tmpCluster$;

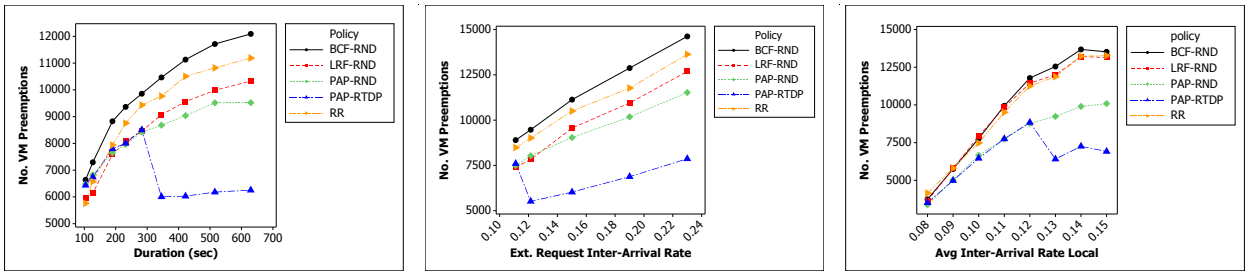
minimum value is chosen (steps 9-16). Finally, Y^i is updated for the selected cluster (step 17).

We examine the efficacy of the proposed policies against following policies. RR: where external requests are distributed based on RR policy. BCF: biggest cluster first. LRF: least rate first. RND also indicates Bernouli (random) dispatch policy.

As we can see in all sub-figures of Figure 4, the number of VMs preempted increases by changing different factors. In all of them PAP-RTDP (combination of proposed workload allocation policy and dispatch policy) significantly outperforms other policies.

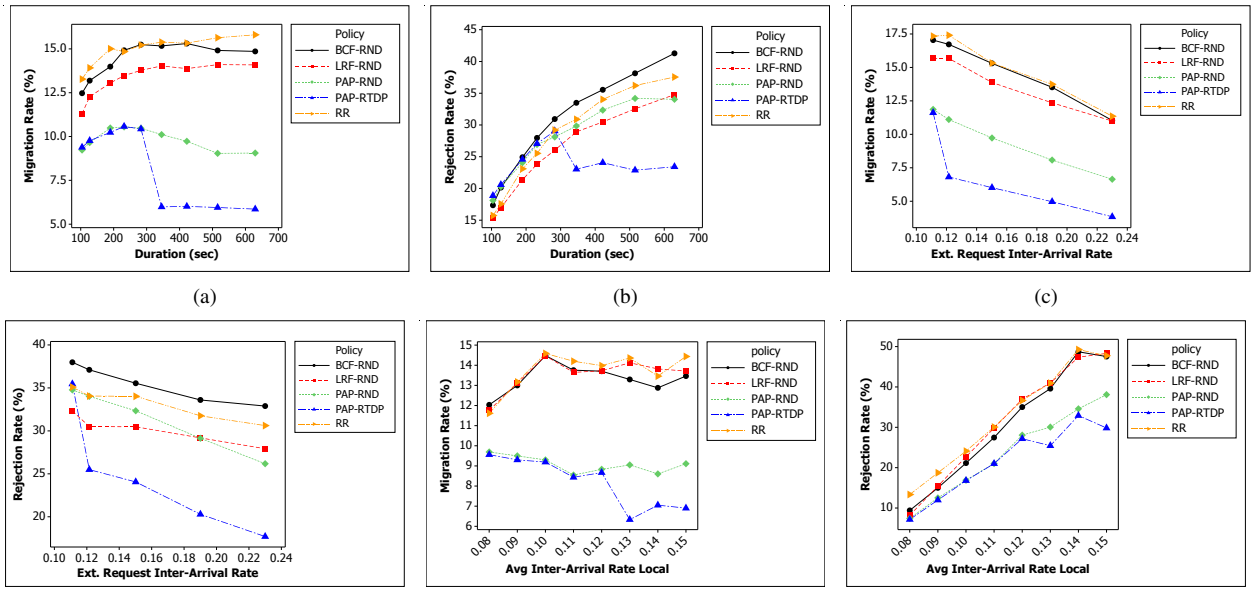
In Figure 4(a), we witness a sharp decrease in PAP-RTDP when the average run time is more than 300 seconds. In fact, in the case of PAP-RTDP, better sequencing of the external requests has resulted in better balance in allocating requests which in turn results in fewer VM preemptions. Figures 4(b) and 4(c) reveal the efficacy of PAP-RND and PAP-RTDP, particularly where the arrival rate of external requests or the arrival rate of local requests are increased. We can conclude that workload allocation policy (PAP) has more impact where inter-arrival rate of local requests is high whereas dispatch policy has more influence where external requests' arrival rate is high.

In the last experiment we measure how valuable (Migratable and Non-preemptive) users are respected. For Migratable requests we measure the number of times that VM migration happens (migration rate). For Non-preemptive external requests we consider the rejection rate. The results of the experiments are illustrated in Figure 5. In all sub-figures of



(a) (b) (c)

Fig. 4. Number of VMs preempted by modifying (a) the average duration, (b) the arrival rate of external requests, and (c) the arrival rate of local requests.



(a) (b) (c)

(d) (e) (f)

Fig. 5. Respecting more valuable users by modifying (a),(b) the average duration, (c),(d) arrival rate of external requests, (e),(f) arrival rate of local requests.

Figure 5, we can notice that PAP-RTDP dispatching policy has substantially reduced the percentage of migrations and also rejections. Particularly, in figure 5(f), although PAP-RTDP is not substantially better than PAP-RND, we observe a marginal improvement in the rejection rate mainly for rates more than 0.12.

In Figures 5(c) and 5(d) as the inter-arrival rate of external requests increases, we observe a decrease in the migration and rejection rates. In fact, by having more external requests the probability of having diverse leases at each time is more. This issue reduces the probability of migration and rejection.

V. Conclusions and Future Work

In this research we considered a virtualized federated Grid environment where local requests and external requests coexist in each cluster and local requests have preemptive priority over the external requests. In this environment, we first explored the appropriate set of leases that can be preempted in a way that both system criteria and user criteria were satisfied. The second part of the research introduces a contention-aware meta-scheduling policy in this environment that reduces the contention between local users and external users. Although we carried out this research in the context of InterGrid, we believe that it is extensively applicable in other Grid/Cloud resource providers where requests with higher priority (such as local or more valuable requests) coexist with other requests. In future, we plan to investigate admission control policies in this environments. Another future direction of this research is considering negotiable QoS requirements for requests.

References

[1] M. Amini Salehi, B. Javadi, and R. Buyya. Performance analysis of preemption-aware scheduling in multi-cluster grid environments. In *ICA3PP*, page In Press, 2011.

[2] M. Amini Salehi, B. Javadi, and R. Buyya. Resource provisioning based on leases preemption in InterGrid. In *Proceeding of the 34th Australasian Computer Science Conference (ACSC 2011)*, pages 25–34, Perth, Australia, 2011.

[3] J. Anselmi and B. Gaujal. Optimal routing in parallel, non-observable queues and the price of anarchy revisited. In *22nd International Teletraffic Congress (ITC)*, Amsterdam, The Netherlands, 2010.

[4] J. S. Chase, D. E. Irwin, L. E. Grit, J. D. Moore, and S. E. Sprenkle. Dynamic virtual clusters in a grid site manager. In *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*, pages 90–98, Washington, DC, USA, 2003.

[5] M. De Assunção, R. Buyya, and S. Venugopal. InterGrid: A case for internetworking islands of Grids. *Concurrency and Computation: Practice and Experience*, 20(8):997–1024, 2008.

[6] B. Sotomayor, K. Keahey, and I. Foster. Combining batch execution and leasing using virtual machines. In *Proceedings of the 17th International Symposium on High Performance Distributed Computing*, pages 87–96, New York, NY, USA, 2008. ACM.