Performance Analysis of Preemption-Aware Scheduling in Multi-cluster Grid Environments

Mohsen Amini Salehi, Bahman Javadi, and Rajkumar Buyya

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

Abstract. In multi-cluster Grids each cluster serves requests from external (Grid) users along with their own local users. The problem arises when there is not sufficient resources for local users (which have high priority) to be served urgently. This problem could be solved by *preempting* resources from Grid users and allocating them to the local users. However, resource preemption entails decreasing resource utilization and increasing Grid users' response time. The question is that how we can minimize the number of preemptions taking place in a resource sharing environment. In this paper, we propose a preemption-aware scheduling policy based on the queuing theory for a virtualized multi-cluster Grid where the number of preemptions is minimized. Simulation results indicate that the proposed scheduling policy significantly decreases the number of virtual machine (VM) preemptions (up to 22.5%).

1 Introduction

Resources provisioning for user applications is one of the main challenges in the Resource sharing environments. Resource sharing environments enable sharing, selection, and aggregation of resources across several Resource Providers (also called clusters in this paper), which are connected through high bandwidth network connections. Nowadays, heavy computational requirements, mostly from scientific communities, are supplied by these resource providers. Examples of production-level resource providers include DAS-2 [5].

Virtual Machine (VM) technology has emerged to enable another style of resource management based on the *lease* abstraction. Due to advantages of this form of management for resource sharing environments [8], we consider a virtualized multi-cluster environment in this paper. Typically, in large-scale resource sharing environments (e.g. InterGrid [3]) computational resources in each cluster are shared between external (Grid) users and local users. Hence, resource provisioning in resource sharing environments is done for two different types of users, namely: local users and Grid users. Local users (hereafter termed local requests), refer to users who ask their local cluster for resources. Grid users (hereafter termed Grid requests) are those users who send their requests to a gateway to get access to larger amount of resources. Typically, local requests have priority over Grid requests in each cluster [3]. In other words, the organization that owns the resources would like to ensure that its community has priority

Y. Xiang et al. (Eds.): ICA3PP 2011, Part I, LNCS 7016, pp. 419–432, 2011.

access to the resources. In this circumstance, Grid requests are welcome to use resources if they are available. Nonetheless, Grid requests should not delay the execution of local requests.

In our previous research [2], we proposed preemption of Grid requests in favor of local requests to remove this contention. We demonstrated that preemption decreases waiting time for local requests. However, the side-effects of preemption is twofold:

- From the system owner perspective, preemption imposes a considerable overhead to the underlying system and degrades resource utilization. This overhead is more notable in circumstances that VMs are used for resource provisioning [8].
- From the Grid user perspective, preemption increases the response time of the Grid requests.

Therefore, the main problem we are dealing with in this research is how to decrease the number of preemptions that take place in a multi-cluster Grid environment.

In this paper, we propose a preemption-aware scheduling policy for a virtualized multi-cluster Grid that distributes Grid requests amongst different clusters in a way that the number of preemptions minimizes. The proposed policy is based on the stochastic analysis of routing in parallel non-observable queues. This policy is not dependent to the availability information of the clusters and does not impose any overhead to the system. In summary our paper makes the following contributions:

- Proposing analytical queuing model based on the routing in parallel non-observable queues.
- Adapting the proposed analytical model to a preemption-aware scheduling policy.
- Evaluating the proposed scheduling policy under realistic workload models.

We consider this problem in the context of InterGrid. In the InterGrid 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 [2].

The rest of this paper is organized as follows: Proposed analytical queuing model is described in Section 2 which is followed by the preemption-aware scheduling policy in Section 3. Performance of the proposed policy is reported in Section 4. Then, in Section 5 related research work are introduced. Finally, conclusion and future works are provided in Section 6.

2 Analytical Queuing Model

In this section we describe the analytical modeling of preemption in a multi-cluster Grid environment based on routing in parallel queues. This section is followed by proposing a scheduling policy in IGG (gateway) built upon the analytical model provided in this part.

The queuing model that represents a gateway along with several non-dedicated clusters (i.e. clusters with shared resources between local and Grid requests) is depicted in Figure. 1. According to this figure, there are N clusters in a Grid where each cluster j receives requests from two independent sources. One source is a stream of local requests with arrival rate λ_j and the other source is a stream of Grid requests which are sent by the gateway with arrival rate $\hat{\Lambda}_j$. The gateway receives Grid requests from other peer gateways [3] (G_1,..,G_g in Figure 1). Therefore, Grid request arrival rate to the gateway is $\Lambda = \bar{\Lambda}_1 + \bar{\Lambda}_2 + ... + \bar{\Lambda}_g$ where g indicates the number of gateways that potentially can send Grid requests to the gateway. Submitted local requests to cluster j must be executed on cluster j unless the requested resources is occupied by another local request or a non-preemptable Grid request. The first and second moment of service time of local requests in cluster j are τ_j and μ_j, respectively. On the other hand, a Grid request can be allocated to any cluster but it might get preempted later on. We consider θ_j and ω_j as the first and second moment of service time of Grid requests on cluster j, respectively. For the sake of clarity, Table 1 gives the list of symbols we use in this paper along with their meaning. Indeed, the analytical model aims at distributing the total original arrival rate of Grid requests (Λ) amongst clusters. In this situation if we consider each cluster as a single queue and the gateway as a meta-scheduler that redirects each incoming Grid request to one of the clusters, then the problem of scheduling Grid requests in the gateway can be considered as a routing problem in distributed parallel queues.

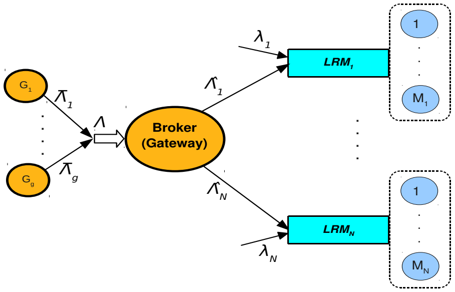


Fig. 1. Queuing model for resource provisioning in a Grid with N clusters

Table 1. Description of symbols used in the queueing model

Symbol	Description
N	Number of clusters
M_j	Number of computing elements in cluster j where $1 \leq j \leq N$
$\bar{\Lambda}_j$	Original arrival rate of Grid requests to cluster j
$\hat{\Lambda}_j$	Arrival rate of Grid requests to cluster j after load distribution
Λ	$= \sum_{i=1}^{g} \bar{\Lambda}_i = \sum_{j=1}^{N} \hat{\Lambda}_j$
θ_j	Average service time of a Grid request on cluster j
ω_j	Second moment of Grid requests service time on cluster j
γ_j	$= \theta_j \cdot \hat{\Lambda}_j$
R_j	Average response time of Grid requests on cluster j
λ_j	Arrival rate of local requests to cluster j
κ_j	Arrival rate of local requests plus Grid requests to cluster j
τ_j	Average service time of local requests on cluster j
μ_j	Second moment of local requests service time on cluster j
ρ_j	$= \tau_j \cdot \lambda_j$
m_j	$= \frac{\hat{\Lambda}_j}{\kappa_j}\omega_j + \frac{\lambda_j}{\kappa_j}\mu_j$
u_j	Utilization of cluster j $(= \gamma_j + \rho_j)$
r_j	Average response time of local requests on cluster j
η_j	Number of VM preemptions that happen in cluster j
T	Average response time of all Grid requests
T_j	Average response time of Grid requests on cluster j

Considering the mentioned situation, the goal of the scheduling in the gateway is to schedule the Grid requests amongst the clusters in a way that minimizes the overall number of VM preemptions in a Grid. Therefore, our primary objective function can be expressed as follows:

$$\min \sum_{j=1}^{N} \eta_j \tag{1}$$

However, minimizing the response time of requests is easier than the number of preemptions in such a system. Furthermore, more researches have been undertaken in similar circumstances to minimize the response time.

The most related research has been carried out by Li [6]. He has analyzed the load distribution problem in a cluster in the presence of two types of requests namely, local (dedicated) and Grid (generic) requests. Nonetheless, Li's goal of optimization is minimizing the response time of Grid (generic) requests whereas our goal is minimizing the overall number of preemptions. The other significant difference is that Li has solved the problem for a single cluster whereas our problem is in the context of a multi-cluster Grid. Li has mentioned the analysis of a multi-cluster system as a future direction of his work. From this perspective, our research can be considered as the future work of Li's research.

Although there are even more differences between our problem and the problem investigated by Li, we believe that this analysis can still be modified and applied to solve our problem. More specifically, from the results of some initial experiments as well as results of our previous research [2] we noticed an association between response time and number of VM preemptions in the system. To assess the strength of association between response time and number of VM preemptions we performed regression analysis between the two factors. Result of the regression analysis shows a positive correlation between number of VM preemptions in a cluster and response time of Grid requests (regression equation:

$R = 3.09 + 0.012\eta$ where R and η indicate the response time of Grid requests and number of VM preemptions). The regression analysis acknowledges that decreasing response time can be also applied for the purpose of minimizing the number of VM preemptions. Details of the modified analysis is discussed over the next paragraphs.

To minimize the average response time of Grid requests we should minimize:

$$T = \frac{1}{\Lambda} \sum_{j=1}^{N} \hat{\Lambda}_j \cdot T_j \tag{2}$$

where the constraint is: $\hat{\Lambda}_1 + \hat{\Lambda}_2 + ... + \hat{\Lambda}_N - \Lambda = 0$. The response time of Grid requests for each cluster j (T_j) is worked out based on Equation 3 by assuming each cluster j as an M/G/1 queue [6]:

$$T_j = \frac{1}{1-\rho_j} \left(\theta_j + \frac{\kappa_j m_j}{2(1-u_j)} \right) \tag{3}$$

Lagrange multiplier method is used to minimize Equation 2. By solving the above minimization problem, input arrival rate of each cluster is calculated based on the Equation 4:

$$\hat{\Lambda}_j = \frac{(1-\rho_j)}{\theta_j} - \frac{1}{\theta_j} \sqrt{\frac{(1-\rho_j)(\omega_j(1-\rho_j)) + \theta_j \lambda_j \mu_j}{2\theta_j(1-\rho_j)z + (\omega_j - 2\theta_j^2)}} \tag{4}$$

where z is the Lagrange multiplier.

Considering that $\Lambda = \hat{\Lambda}_1 + \hat{\Lambda}_1 + ... + \hat{\Lambda}_N$, then z can be calculated using the following Equation:

$$\begin{aligned} \sum_{j=1}^{N} \frac{1}{\theta_j} \sqrt{\frac{(1-\rho_j)(\omega_j(1-\rho_j)) + \theta_j \lambda_j \mu_j}{2\theta_j(1-\rho_j)z + (\omega_j - 2\theta_j^2)}} \\ = \left(\sum_{j=1}^{N} \frac{(1-\rho_j)}{\theta_j} \right) - \Lambda \end{aligned} \tag{5}$$

In fact, Equation 5 expresses the relation between different parameters of the system in which z is unknown. By solving Equation 5 for all clusters and working out z, Equation 4 can be solved. However, finding a generic closed form solution in Equation 5 for finding z is impossible. Nonetheless, a numerical solution can be found by searching z in range of $[lb,ub]$ using a bisection algorithm [6]. For this purpose, considering that $\hat{\Lambda}_j \geq 0$ and from Equation 4 we can infer that:

$$z \geq \frac{\lambda_j \mu_j}{2(1-\rho_j)^2} + \frac{\theta_j}{(1-\rho_j)} \tag{6}$$

Therefore, for all $1 \leq j \leq N$ the lower bound (lb) of the interval is:

$$lb = \max_{j=1}^{N} \left(\frac{\lambda_j \mu_j}{2(1-\rho_j)^2} + \frac{\theta_j}{(1-\rho_j)} \right) \tag{7}$$

If we define $\phi_j(z)$ according to Equation 8:

$$\phi_j(z) = \frac{1}{\theta_j}\sqrt{\frac{(1-\rho_j)(\omega_j(1-\rho_j)) + \theta_j\lambda_j\mu_j}{2\theta_j(1-\rho_j)z + (\omega_j - 2\theta_j^2)}} \tag{8}$$

and considering Equation 5, then we have:

$$\sum_{j=1}^{N} \phi_j(lb) \geq \left(\sum_{j=1}^{N} \frac{(1-\rho_j)}{\theta_j}\right) - \Lambda \tag{9}$$

The upper bound also can be worked out based on Equation 10. ub can be reached by doubling lb up until the condition is met.

$$\sum_{j=1}^{N} \phi_j(ub) \leq \left(\sum_{j=1}^{N} \frac{(1-\rho_j)}{\theta_j}\right) - \Lambda \tag{10}$$

If condition in Equation 9 is not met, then we have to decrease lb by removing clusters which are heavily loaded. Load of a cluster j is comprised of local requests that have been arrived and Grid requests which are already assigned to the cluster. The load can be calculated as follows.

$$\psi_j = \frac{\lambda_j\mu_j}{2(1-\rho_j)^2} + \frac{\theta_j}{(1-\rho_j)} \tag{11}$$

For the sake of simplicity, in Equation 12 we have assumed that $\psi_1 \leq \psi_2 ... \leq \psi_N$.

$$\sum_{j=1}^{k} \phi_j(\psi_k) \geq \left(\sum_{j=1}^{k} \frac{(1-\rho_j)}{\theta_j}\right) - \Lambda \tag{12}$$

It is worth mentioning that values bigger than k would not receive any Grid request from the gateway (i.e. $\hat{\Lambda}_{k+1} = \hat{\Lambda}_{k+2} = ... = \hat{\Lambda}_N = 0$).

3 Preemption-Aware Scheduling Policy

In this section we discuss how the analysis mentioned in previous section can be adapted as the scheduling policy for Grid requests inside IGG.

In fact, the analysis provided in Section 2 was based on some widely used assumptions. However, some of these assumptions do not hold for case of the multi-cluster that we are investigating. In the analysis we assumed that:

- each cluster was an M/G/1 queue. However, in InterGrid we are investigating each cluster as a G/G/M_j queue.
- all requests needed one VM. However, in InterGrid we consider requests that need several VMs for a certain amount of time.
- local requests could preempt Grid requests. However, in InterGrid not all Grid requests are preemptable. In fact, if the Grid request is *Non-Preemptable*, it cannot be preempted by local requests.

- each queue is run in FCFS fashion. However, in order to improve the resource utilization we consider conservative backfilling method in the local schedulers.

Considering the above differences, we do not expect that the preemption-aware scheduling policy performs optimally. In fact, we are trying to examine how efficient the above analysis would be by substituting the above assumptions with some approximations.

To adapt the analysis in a way that covers requests that need several VMs we modify the service time of Grid requests on cluster j (θ_j) and local requests on cluster j (τ_j) in the following way:

$$\theta_j = \frac{\bar{\upsilon}_j \cdot \bar{d}_j}{M_j s_j} \tag{13}$$

$$\tau_j = \frac{\bar{\zeta}_j \cdot \bar{\varepsilon}_j}{M_j s_j} \tag{14}$$

where $\bar{\upsilon}_j$ and $\bar{d}_j$ show the average number of VMs needed and average duration of Grid requests. $\bar{\zeta}_j$ and $\bar{\varepsilon}_j$ show the average number of VMs needed and average duration of local requests. Finally, s_j shows the processing speed in cluster j. This change also affects second moment of service time for both local and Grid requests. We can use coefficient of variance ($CV = StDev/Mean$) to obtain the modified second moment. Assuming that CV is given, the second moment of service time for Grid and local requests on cluster j is calculated according to Equation 15 and 16, respectively.

$$\omega_j = (\alpha_j \cdot \theta_j)^2 + \theta_j^2 \tag{15}$$

$$\mu_j = (\beta_j \cdot \tau_j)^2 + \tau_j^2 \tag{16}$$

where α_j and β_j show the CV of Grid requests and local requests service time on cluster j respectively. The preemption-aware scheduling policy (PAP), which is built upon analysis of Section 2, is shown in the form of pseudo-code in Algorithm 1. According to Algorithm 1, at first ψ is calculated for all clusters. Then, in steps 3 to 9, to exclude the heavily loaded clusters, clusters are sorted based on the ψ value in the ascending order. Then, the value of k is increased up until condition defined in Equation 12 (step 6) is met. ub is found by starting from $2 \cdot lb$ and is doubled up until condition in step 12 is met. Steps 14-19 show the bisection algorithm mentioned in Section 2 for finding proper value for z. Finally, in steps 20 and 21 the arrival rate to each cluster is determined. Steps 22 and 23 guarantee that clusters $k + 1$ to N, which are heavily loaded, do not receive any Grid request.

4 Performance Evaluation

4.1 Experimental Setup

We use GridSim a discrete event simulator, to evaluate performance of the scheduling policies. We consider a Grid with 3 clusters with 32, 64, and 128

Algorithm 1. Preemption-Aware Scheduling Policy (PAP)

```
Input: $\bar{\Lambda}_j$,$\theta_j$,$\omega_j$,$\lambda_j$,$\tau_j$,$\mu_j$, for all $1 \leq j \leq N$.
Output: ($\hat{\Lambda}_j$) load distribution of Grid requests to different clusters, for all
        $1 \leq j \leq N$.
for $j \leftarrow 1$ to $N$ do
    $\psi_j = \frac{\lambda_j \mu_j}{2(1-\rho_j)^2} + \frac{\theta_j}{(1-\rho_j)}$;
Sort ($\psi$);
$k \leftarrow 1$;
while $k < N$ do
    if $\sum_{j=1}^{k} \phi_j(\psi_k) \geq \left( \sum_{j=1}^{k} \frac{(1-\rho_j)}{\theta_j} \right) - \Lambda$ then
        break;
    else
        $k \leftarrow k+1$;
$lb \leftarrow \psi_k$;
$ub = 2 * lb$;
while $\sum_{j=1}^{k} \phi_j(ub) > \left( \sum_{j=1}^{k} \frac{(1-\rho_j)}{\theta_j} \right) - \Lambda$ do
    $ub = 2 * ub$;
while $ub - lb > \epsilon$ do
    $z \leftarrow (lb + ub)/2$;
    if $\sum_{j=1}^{k} \phi_j(z) \geq \left( \sum_{j=1}^{k} \frac{(1-\rho_j)}{\theta_j} \right) - \Lambda$ then
        $lb \leftarrow z$;
    else
        $ub \leftarrow z$;
for $j \leftarrow 1$ to $k$ do
    $\hat{\Lambda}_j = \frac{(1-\rho_j)}{\theta_j} - \frac{1}{\theta_j}\sqrt{\frac{(1-\rho_j)(\omega_j(1-\rho_j))+\theta_j\lambda_j\mu_j}{2\theta_j(1-\rho_j)z+(\omega_j-2\theta_j^2)}}$;
for $j \leftarrow k+1$ to $N$ do
    $\hat{\Lambda}_j = 0$;
```

nodes with homogeneous computing speed $s_j = 1000$ MIPS for all clusters. Each cluster is managed by an LRM and a conservative backfilling scheduler. Clusters are interconnected using a 1000 Mbps network bandwidth. We assume all nodes of each cluster as a single core with one VM. The maximum number of VMs in the generated requests of each cluster does not exceed the number of nodes in that cluster. We consider size of each VM, 1024 MB [10]. The overhead time imposed by preempting VMs varies based on the type of Grid leases involved in preemption [8]. For *Cancelable* leases the overhead is the time needed to terminate the lease and shutdown its VMs. This time is usually much lower than the time needed for suspending or migrating leases [8]. In our experiments, suspension time (t_s) and resumption time (t_r) are 160 and 126 seconds, respectively [8]. The time overhead for transferring (migrating) a VM with similar configuration is 165 seconds [10].

Baseline Policies. For the sake of comparison, we evaluate the proposed scheduling policy (PAP) against other two policies which are described below:

- Round Robin Policy (RRP): In this policy IGG distributes Grid requests between different clusters of a Grid in a round-robin fashion with a deterministic sequence. Formally, this policy is demonstrated as $\hat{\Lambda}_j = \Lambda / N$
- Least Rate Policy (LRP): In this policy the rate of Grid requests submitted to each cluster has inverse relation with arrival rate of local requests to that cluster. In other words, clusters that have larger rate of incoming local requests would be assigned less number of Grid requests by IGG. Formal presentation of the policy is as $\hat{\Lambda}_j = (1 - \frac{\lambda_j}{\sum_{j=1}^{N} \lambda_j}) \cdot \Lambda$

We have also implemented PAP with the following details:

- We assumed that in step 14 of Algorithm 1 the precision is 1 ($\epsilon = 1$).
- In Equations 15 and 16, to work out the second moment of service time for local and Grid requests, we assumed that in all clusters $\alpha_j = \beta_j = 1$ (i.e. CV of service time for both Grid and local requests is 1).
- We believe that users mostly request for Suspendable and Nonpreemptable types. Therefore, in the experiments we consider: BE-Suspendable:40%; BE-Cancelable:10%; DC-Nonpreemptable:40%; and DC-Migratable:10%. These request types are uniformly distributed in Grid requests.

Workload Model. In the experiments conducted, DAS-2 workload model [5] has been configured to generate two-day-long workload of parallel requests. This workload model is based on the DAS-2 multi-cluster in the Netherlands.

We intend to study the behavior of different policies when they face workloads with different characteristics. More specifically, we study situations where Grid requests have:

- different number of requested VMs: In this case for Grid requests, we keep average *duration=30* minutes and average *arrival rate=1.0.*
- different request duration: In this case for Grid requests, we keep average *number of VMs=3.0* and average *arrival rate=1.0.*
- different arrival rate: In this case for Grid requests, we keep average *number of VMs=3.0* and average *duration=30* minutes.

Each experiment is performed on each of these workloads separately for 30 times and the average of the results is reported. To generate these workloads, we modify parameters of DAS-2 model. Local and Grid requests have different distributions in each cluster. Based on the workload characterization [5], the inter-arrival time, request size, and request duration follow Weibull, two-stage Loguniform, and Lognormal distributions, respectively. These distributions with their parameters are listed in Table 2.

4.2 Experimental Results

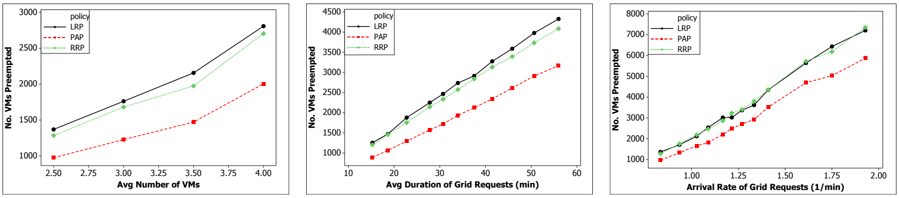
Number of VM Preemptions. As mentioned earlier, both resource owners and users benefit from fewer VM preemptions. From the resource owner

Table 2. Input parameters for the workload model

Input Parameter	Distribution	Values Grid Requests	Values Local Requests
No. of VMs	Loguniform	$(l = 0.8, 1.5 \leq m \leq 3, h = 5, q = 0.9)$	$(l = 0.8, m = 3, h = 5, q = 0.9)$
Request Duration	Lognormal	$(1.5 \leq a \leq 2.6, b = 1.5)$	$(a = 1.5, b = 1.0)$
Inter-arrival Time	Weibull	$(0.7 \leq \alpha \leq 3, \beta = 0.5)$	$(\alpha = 0.7, \beta = 0.4)$
P_{one}	N/A	0.2	0.3
P_{pow2}	N/A	0.5	0.6

perspective, fewer preemption leads to less overhead for the underlying system and improves the utilization of resources. From the user perspective, however, preempting Grid leases has different impacts based on the lease types. For Suspendable and Migratable leases, preemption leads to increasing completion time. For Cancelable leases preemption results in terminating that lease. Since users of different lease types have distinct expectation from the system, it is not easy to propose a common criterion to measure user satisfaction. Nonetheless, all types of leases Grid users suffer from lease preemption. Therefore, we believe that the number of VM preemptions in a Grid is a generic enough metric to express Grid users' satisfaction. In this experiment we report the number of VMs getting preempted by applying different scheduling policies. As we can see in all sub-figures of Figure 2, the number of VMs preempted almost linearly increases by increasing the average number of VMs (Figure 2(a)), duration (Figure 2(b)), and arrival rate of Grid requests (Figure 2(c)).

In all cases PAP outperforms other policies specially when the average number of VMs increases or when duration of Grid requests increases. Nonetheless, we observe less difference between the PAP and two other policies when the inter-arrival time of Grid requests increases (Figure 2(c)). In all cases the difference between PAP and other policies become more significant when there is more load in the system which shows the efficiency of PAP when the system is heavily loaded. In the best situation (in Figure 2(b) where the average duration of Grid requests is 55 minutes) we observe that PAP results in around 1000 (22.5%) less VM preemptions.



(a) (b) (c)

Fig. 2. Number of VMs preempted by applying different policies. By modifying (a) the average number of VMs, (b) the average duration, and (c) the arrival rate of Grid requests.

Resource Utilization. Time overhead due to VM preemptions leads to resource under-utilization. Therefore, we are interested to see how different scheduling policies affect the resource utilization. Resource utilization is defined as follows:

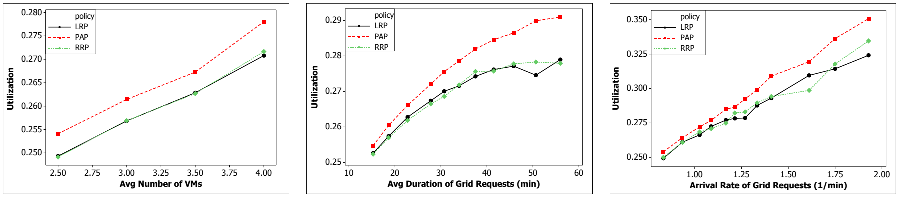
$$Utilization = \frac{computationTime}{totalTime} \tag{17}$$

where:

$$computationTime = \sum_{i=1}^{|L|} v(l_i) \cdot d(l_i) \tag{18}$$

where $|L|$ is the total number of leases allocated, $v(l_i)$ is the number of VMs in lease l_i, $d(l_i)$ is the duration of lease l_i.

In this experiment we explore the impact of preempting VMs on the resource utilization as a system centric metric. In general, resource utilization resulted from applying PAP is better than other policies as depicted in Figure 3. However, the difference is more remarkable when the average number of VMs or arrival rate of Grid requests increases (Figures 3(b) and 3(c)). We observe that PAP, which causes fewer preemptions, results in better resource utilization. In Figure 3(b), we can see that in all policies resource utilization becomes almost flat when Grid requests become long (more than 40 minutes). The reason is that when requests become long, the useful computation time dominates the overhead of VM preemptions. We can infer that VM preemption does not significantly affect resource utilization when requests are long (more than 40 minutes).



(a) (b) (c)

Fig. 3. Resource utilization resulted from different policies. By modifying (a) the average number of VMs, (b) the average duration, and (c) the arrival rate of Grid requests.

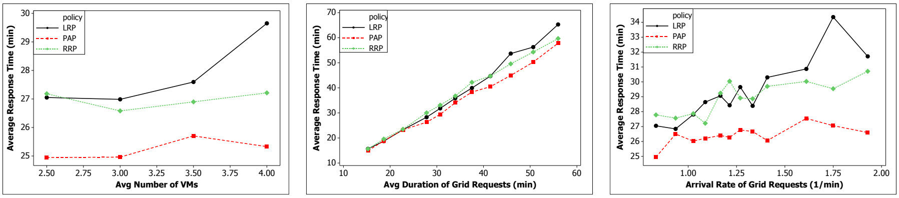
Average Response Time (ART). We are interested in ART metric to see how the investigated scheduling policies affect response time of Best-Effort Grid requests. In fact, this metric measures the amount of time on average a Best-Effort lease should wait beyond its ready time to get completed. ART in each cluster is calculated based on the Equation 19.

$$ART_j = \frac{\sum_{l \in \Delta}(c_l - b_l)}{|\Delta|} \tag{19}$$

where Δ is the set of Best-Effort leases. c_l and b_l show completion time and ready time for lease l, respectively. Then, ART over all clusters is the weighted average ART in each cluster.

According to the results in Figure 4, we conclude that PAP results in better ART for Grid requests. However, unlike the previous experiments, the response time does not decrease significantly when the duration of the Grid requests increased (Figure 4(b)). The reason is that when the requests become longer, the duration and waiting times of requests normally become more dominant factor in response time comparing with the waiting times imposed because of preemption. Therefore, the number of VM preemptions is not significantly effective on average response time of the leases, particularly, when the average duration of leases is long.

We also conclude that ART does not change significantly by increasing the average number of VMs in the Grid requests (Figure 4(a) after 3.5) or their inter-arrival time (Figure 4(c) after 1.6). In fact in both cases by increasing average number of VMs of the Grid requests or their inter-arrival, more Deadline-Constraint Grid requests and even more local requests get rejected. This makes more places for other requests to fit in. Therefore, ART does not increase or even slightly decrease. For instance, in Figure 4(c), where the arrival rate for Grid requests is more than 1.6, we experience 13.5% improvement in ART.



(a) (b) (c)

Fig. 4. Average response time resulted from different policies. By modifying (a) the average number of VMs, (b) the average duration, and (c) the arrival rate of Grid requests.

5 Related Work

Assuncao et al. [3] have proposed adaptive partitioning of the availability times between local and Grid requests in each cluster. Each cluster submits its availability information to the IGG periodically. Therefore, there is a communication overhead between IGG and clusters for submitting availability information. Hence, there is a possibility that the availability information be imprecise.

Huedo et al. [9] have investigated the usage of multiple meta-schedulers to make loosely coupled connection between Grids. They use Gridway to migrate jobs from a remote cluster when the job does not get the expected processing power. However, they do not discuss how we can prioritize organization level requests versus requests coming from other Grids.

Haizea [8] is a lease scheduler which schedules a combination of advanced reservation and best effort leases. Haizea preempts best effort leases in favor of advance reservation requests. Sotomayor et al. [8], have also investigated the overhead time imposed by preempting a lease in Haizea. By contrast, we propose a scheduling policy to decrease the number of preemptions in the system.

Scojo-PECT [7] is a preemptive scheduler that aims at making a fair share scheduling between different job classes of a Grid. The approach is applying coarse-grain time sharing and suspending VMs on disk. However, the authors do not consider the overhead of suspending VMs on disk in their evaluations. The main difference with our work is the goal of scheduling. We minimize the number of VM preemptions whereas Sodan et.al's goal is fair share scheduling.

Amar et al. [1] have added preemption to cope with the non-optimality in on-line scheduling policies. The preemption policy prioritize jobs based on their remaining time as well as the job's weight. Our research is different with this work in the sense that they do not consider the lease based resource provisioning. Moreover, we try to minimize the number of preemption in a Grid where several types of Grid requests coexist.

Kettimuthu et al. [4] proposed a preemption policy, which is called Selective Suspension, where an idle job can preempt a running job if the suspension factor is adequately more than running job. The authors do not specify how to minimize the number of preemptions, instead, they decide when to do the preemption.

6 Conclusions and Future Work

In this research we proposed a preemption-aware scheduling policy (PAP) in IGG, as a virtualized multi-cluster resource sharing environment, that minimizes the side-effects of VM preemptions. Experimental results indicate that PAP resulted in up to 1000 less VM preemptions (22.5% improvement) comparing with other policies in a two-day-long workload. This decrease in number of VM preemptions improves the utilization of the resources and decreases average response time of the Grid requests (up to 13.5%). We believe that our policy is extensively applicable in lease-based Grid/Cloud resource providers where requests with higher priority coexist with other requests. A nice application is in Cloud (IaaS) providers where there is certain priorities between different users; and resource owners tend to minimize the number of VM preemptions. In future we plan to investigate how IGG can consider deadline and other QoS issues in its scheduling. Another extension would be considering co-allocation of the incoming Grid requests on different clusters to further decrease the number of preemptions.

References

1. Amar, L., Mu'alem, A., Stößer, J.: The power of preemption in economic online markets. In: Altmann, J., Neumann, D., Fahringer, T. (eds.) GECON 2008. LNCS, vol. 5206, pp. 41–57. Springer, Heidelberg (2008)
2. Amini Salehi, M., Javadi, B., Buyya, R.: Resource provisioning based on leases preemption in intergrid. In: Proceeding of the 34th Australasian Computer Science Conference (ACSC 2011), Perth, Australia, pp. 25–34 (2011)
3. de Assunção, M.D., Buyya, R.: Performance analysis of multiple site resource provisioning: Effects of the precision of availability information. In: Sadayappan, P., Parashar, M., Badrinath, R., Prasanna, V.K. (eds.) HiPC 2008. LNCS, vol. 5374, pp. 157–168. Springer, Heidelberg (2008)

4. Kettimuthu, R., Subramani, V., Srinivasan, S., Gopalsamy, T., Panda, D.K., Sadayappan, P.: Selective preemption strategies for parallel job scheduling. Intl. Journal of High Performance Computing and Networking 3(2/3), 122–152 (2005)
5. Li, H., Groep, D.L., Wolters, L.: Workload characteristics of a multi-cluster supercomputer. In: Feitelson, D.G., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2004. LNCS, vol. 3277, pp. 176–193. Springer, Heidelberg (2005)
6. Li, K.: Optimal load distribution in nondedicated heterogeneous cluster and grid computing environments. J. System Architecture 54, 111–123 (2008)
7. Sodan, A.: Service control with the preemptive parallel job scheduler scojo-pect. Journal of Cluster Computing, 1–18 (2010)
8. Sotomayor, B., Keahey, K., Foster, I.: Combining batch execution and leasing using virtual machines. In: Proceedings of the 17th International Symposium on High Performance Distributed Computing, New York, NY, USA, pp. 87–96 (2008)
9. Vázquez-Poletti, J.L., Huedo, E., Montero, R.S., Llorente, I.M.: A comparison between two grid scheduling philosophies: Egee wms and grid way. Multiagent Grid Syst. 3, 429–439 (2007)
10. Zhao, M., Figueiredo, R.: Experimental study of virtual machine migration in support of reservation of cluster resources. In: Proceedings of the 3rd International Workshop on Virtualization Technology in Distributed Computing, pp. 5–11. ACM, New York (2007)