# Resource Provisioning based on Lease Preemption in InterGrid

**Mohsen Amini Salehi** **Bahman Javadi** **Rajkumar Buyya**

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

**Abstract**

Resource provisioning is one of the main challenges in resource sharing environments such as InterGrid. Recently, many resource management systems in resource sharing environments use lease abstraction and virtual machines for provisioning. In resource sharing environments resource providers serve 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 higher priority than grid users, and need resources urgently. This problem could be solved by preempting leases from grid users and allocating them to the local users. However, preempting leases entails determining which lease(s) are better choices to be preempted and what should be done with the preempted leases. To answer these questions, in this work, we propose different request types in the InterGrid environment. Then, we propose and compare several policies that determine the proper set of lease(s) for preemption. The first policy increases resource utilization as a system centric criterion. The second policy improves user satisfaction by decreasing the number of preempted leases. The third policy makes a trade-off between resource utilization and the number of lease preemption. Simulation results demonstrate that the proposed preemption policies serve up to 72% more local requests without increasing the rejection ratio of grid requests.

## 1 Introduction

Managing and providing computational resources for user applications is one of the challenges in the high performance computing community. Resource sharing environments enable sharing, selection, and aggregation of different resources across several Resource Providers (RP), which are also called sites, and usually scattered over a geographical region. These RPs are connected through high bandwidth network connections. Nowadays, heavy computational requirements, mostly from scientific communities, are supplied by these RPs, such as Grid 5000 in France and DAS-2 in the Netherlands.

InterGrid (De Assunção et al. 2008), provides an architecture and policies for inter-connecting different Grids. As shown in Figure 1, in InterGrid computational resources in each RP are shared between grid users as well as local users. The provisioning rights over the resources from several RPs inside a Grid are delegated to the InterGrid Gateway (IGG). IGGs coordinate resource allocation for requests through predefined contracts between Grids (De Assunção et al. 2008). On the other hand, local users send their requests directly to the local scheduler of the RP.

Hence, resource provisioning in InterGrid is done for two different types of users, namely: local users and external (grid) users. As illustrated in Figure 1, local users (hereafter termed local request), refer to users who ask their local RP for resources. Grid users (hereafter termed grid request) are those users who send their requests to the IGG to get access to larger amount of resources. Typically, for an RP local requests have more priority than grid requests (Chase et al. 2003). However, removing the contention between the local request and grid request is challenging. In other words, the organization that owns the resources would like to ensure that its community has priority 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.

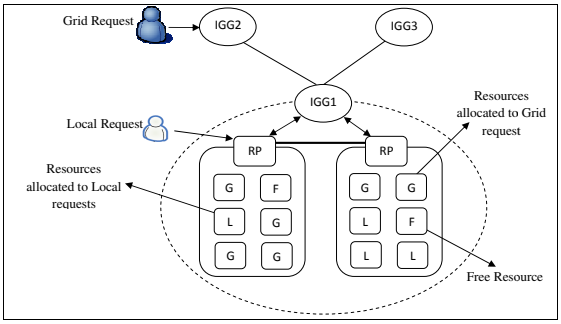


Figure 1: A scenario that shows the contention between local and grid requests in InterGrid.

In InterGrid the resource provisioning is based on the *lease* abstraction. A lease is an agreement between resource provider and resource consumer whereby the provider agrees to allocate resources to the consumer according to the lease terms presented by the consumer (Sotomayor et al. 2008, 2009). Virtual Machine (VM) technology is a way to implement lease-based resource provisioning (Sotomayor et al. 2009). The capabilities of VMs in getting suspended, resumed, stopped, or even migrated (when there is enough bandwidth) have been extensively studied and have shown to be useful in resource provisioning without major utilization loss (Chase et al. 2003, Sotomayor et al. 2008, Zhao & Figueiredo 2007). InterGrid makes one lease for each user VM request.

In this paper, we solve the problem of contention between local and grid requests in InterGrid. Given the fact that local requests have more priority rather than local requests we preempt grid leases in favor

of local requests. More specifically, we propose three policies to determine an appropriate set of leases for preemption. Finally, we make an appropriate decision for the preempted lease.

The rest of this paper is organized as follows: In Section 2, the problem we are investigating is discussed, then in Section 3 related research works are introduced. Proposed policies are described in Section 4 and detailed evaluations are mentioned in Section 5. Finally, conclusion and future works are provided in Section 6.

## 2 Problem Statement

As mentioned earlier, local requests have higher priority than grid requests in InterGrid. The problem we are dealing with in this paper is resource provisioning for local requests when existing resources have been allocated to grid requests and the rest of resources are not adequate to serve the local requests. In this situation, one solution is to preempt some of the leases allocated to grid users. However, there are some challenges in preempting leases.

The first challenge is that there are some restrictions in preempting leases. In fact, one difference between job-based resource provisioning and lease-based resource provisioning is that jobs can be preempted without notifying the user (job owner). Nevertheless, this is not the case for leases (Grit et al. 2007). Therefore, the first challenge is coming up with regulations in the lease terms to make lease preemption possible.

Moreover, making a proper decision for a preempted lease, given the characteristics of a resource sharing environment such as InterGrid, is challenging.

Since lease preemption has some side-effects, the second challenge we consider is how to minimize these side-effects. These side-effects and the challenge ahead are discussed over the next paragraphs.

In the case of job-based resource provisioning, many distributed systems do not provide the ability of preempting jobs (Snell et al. 2002). This is mainly because the operating system has to provide the security of not accessing files and data of the preempted processes. Additionally, since the operating systems mostly do not provide the checkpointing facilities, the preempted jobs have to be killed, which is a waste of resources (Snell et al. 2002). These problems are obviated by leveraging VMs in lease-based resource provisioning (Sotomayor et al. 2008). However, preempting leases and subsequently VMs is not free of cost and imposes time overhead to the system which is one of the side-effects of preemption.

Since the next time-slots are already reserved for other requests, preempting leases and allocating to new requests can potentially affect these reservations. Re-scheduling preempted leases and affected reservations are also side-effects of preempting leases. Preempting leases also makes grid users wait for a longer time to get their leases completed.

By getting lease preemption possible in an RP, there is a possibility that several leases have to be preempted to make sufficient vacant resources for a local request. Therefore, there are potentially several sets of candidate leases that can be preempted. In this paper each set of the candidate leases is termed a *Candidate Set*. However, selecting different candidate sets affects the amount of imposed overhead as well as the number of grid users who get affected by preemption.

These issues bring another challenge into the picture. The challenge is choosing the optimal candidate set for preemption in a way that minimizes the side-effects of preempting leases.

In summary, there are two main challenges ahead:

1. How to make lease preemption possible and what are the potential decisions for preempted leases?
2. Which candidate set is a better choice for preemption? (Preemption policy)
3. We evaluate the impact of the proposed policies upon key performance metrics in a simulation environment.

Formal definition of the problem can be stated as follows:

- $L_i$: Lease $i$.
- $R_j$: Local request $j$
- $\tau(L_i) \in \{gridCancelable, gridSuspendable, gridMigratable, gridNonPreemptable, localNonPreemptable\}$
- $v(x_i)$: Number of VMs in the lease/request $i$.
- $h(L_i)$: Overhead of preempting lease $i$.
- $p(L_i)$: Category of lease $i$ (local or grid) and defined as follows:

$$p(L_i) = \begin{cases} 1 & \text{if grid request} \\ 0 & \text{if local request} \end{cases}$$

According to the above definitions, candidate set $m$ for allocating request $R_j$ can be presented as follows:

$$C_m : \{\forall L_i \,|\, p(L_i) = 1 \;\&\; v(R_j) \leq \sum_{i=1}^{N_m} v(L_i)\} \qquad (1)$$

where $N_m$ is the number of leases involved in candidate set $C_m$. If there are $S$ candidate sets, then all candidate sets can be peresented as:

$$A : \{C_m \,|\, 0 \leq m \leq S-1\} \qquad (2)$$

Finally, a preemption policy can be presented as a function that selects an appropriate candidate set out of all candidate sets (i.e., $policy(A) = C_m$).

We believe that in an RP with many nodes and requests, selecting a proper candidate set for preemption is crucial and leads to significant reduction in preemption overhead time and increases user satisfaction. Although the problem we are investigating in this paper is in InterGrid context, it could be also applied to other lease-based Grid/Cloud resource providers where requests with higher priority (such as local or organisational requests) coexist with other requests.

## 3 Related Work

Although preemption was not extensively studied in distributed computing previously (Snell et al. 2002), recently many researches have been undertaken in the area.

Haizea (Sotomayor et al. 2008) is a lease scheduler which schedules advanced reservation and best effort leases. Haizea preempts best effort leases in favor of advance reservation requests. In case of preempting a lease, Haizea just considers the preemptability of the lease. In other words, when there are several candidate sets to be preempted, Haizea does not have any specific policy to determine which candidate set is a better choice for preemption. In contrast, we propose and compare policies that determine which candidate set is a better choice for preemption. In Haizea, preempted leases are suspended and put in the queue to

be resumed in another available time-slot later. However, we consider a diversity of leases (Cancelable, Suspendable, Migratable, and non-preemptable) that gives the scheduler more options than just suspending the lease. Sotomayor et al. (Sotomayor et al. 2008, 2009) have directly mentioned that lease requests should be categorized and decided based on user-provided priorities. Therefore, our work can be considered as complementary for the research undertaken by Sotomayor et al..

In another research undertaken by Sotomayor et al. (Sotomayor et al. 2009), the overhead time imposed by preempting a lease (suspending and resuming a VM) is estimated. We use the same model to consider the overhead in our evaluations. This overhead should be taken into consideration in lease scheduling where some leases should be preempted in favor of other requests. The proposed model is based on the amount of memory that should be de-allocated, number of VMs mapped to each physical node, local or global memory used for allocating VMs, and the delay related to commands being enacted. In evaluation of the proposed preempting policies, we consider the overhead involved in preempting leases based on this model.

Walters et al. (Walters et al. 2008) introduced a preemption-based scheduling policy for batch and interactive jobs inside a cluster. In this work batch jobs are preempted in favor of interactive jobs. The authors introduce different challenges in preempting jobs including selecting a proper job to be preempted, how to checkpoint the preempted job, how to provision VMs, and how to resume the preempted job. Their preemption policy is based on weighted summation of several factors such as the time spent in the queue.

One difference of our work with Walters et al. is that our preemption policy is based on lease based resource provisioning, while Walter's research is based on job. Moreover, Walters et al. do not consider the circumstance that several jobs should be preempted to make room for the higher priority jobs. Another difference is that Walters et al. do not consider the impact of preemption on reservations in the queue. By contrast, our work considers both the running leases as well as reservations in the queue.

Kettimuthu et al. (Kettimuthu et al. 2005) focused on the impact of preempting parallel jobs in supercomputers for improving the average and worst case slow down of jobs. The authors also suggested 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.

However, the authors do not specify which job should be preempted instead they decide when to do the preemption. The proposed policy is starvation free since it operates based on the response ratio of jobs.

Isard et al. (Isard et al. 2009) investigate the problem of optimal scheduling for data intensive applications such as Map-Reduce (Isard et al. 2009) on the clusters that computing and storage resources are close together. This work provides an example of Cancelable leases that can be terminated without any notification to the job owner. Achieving the optimal resource allocation, they propose a scheduling policy that preempts the currently running job in order to maintain data locality for a new job. Although the scheduling policy is based on job preemption, the authors do not discuss which job is selected to be preempted amongst several candidates.

Snell et al. (Snell et al. 2002) consider the impact of preemption on backfilling scheduling. They provide policies to select the set of jobs for preemption in a way that the jobs with higher priority jobs are satisfied and at the same time the utilization of resources increase. In this work the preempted job is restarted and rescheduled in the next available time-slot. Our work is different with Snell et al. from several aspects. Firstly, since we consider lease based resource provisioning, there are more choices for the preempted lease (such as suspended, migrated, etc). Secondly, Snell et al. recognize the best set of running jobs for preemption. However, in our contribution the preemption policy considers the best candidate set as well as the impact of preempting on the reservations made for the leases in the queue. The third difference is that Snell et al. does not consider the overhead of preempting jobs. In fact, by killing the preempted jobs they reduced the overhead to zero. Nonetheless, the computational power is wasted in that case.

## 4 Proposed Solution

### 4.1 Introducing Different Lease Types

To tackle challenges mentioned in Section 2, we should first make the preemption possible in lease terms agreed between resource provider and consumer. For that purpose, we introduce different request types in InterGrid. After reservation is done for a request, the "request type" is mapped to "lease type".

At the moment, a request issued by a user in InterGrid is composed of the following characteristics:

- Virtual Machine (VM) name needed by the user.
- Number of VMs needed.
- Ready time: the time that requested VMs should be ready.
- Wall time: duration of the lease.
- Deadline: the time that serving the request must be finished.

We extend the InterGrid request by adding the "request type" to it.

Considering the characteristics of a resource sharing environment, proposed request (lease) types give more choices to the scheduler rather than just suspending and rescheduling the preempted leases (Sotomayor et al. 2008). Based on the lease type, the scheduler determines how to schedule the lease and what to be done with a preempted lease.

Different request types we consider for requests in InterGrid are broadly classified as best effort and deadline constraint requests. More details of different request types are as follows:

- *Best Effort-Cancelable*: these requests can be scheduled at any time after their ready time. Leases of such type can be canceled without notifying the lease owner. Cancelable leases neither guarantee the deadline nor the wall time of the lease. Such leases are suitable for map-reduce-like requests (Isard et al. 2009). Spot instances in Amazon EC2[1] are also another example of Cancelable leases. Cancelable leases impose the minimum overhead time at the time of preemption. This overhead is related to the time needed to terminate VMs allocated to the lease.
- *Best Effort-Suspendable*: leases of this type can be suspended at any time but should be resumed later. This type of lease guarantees the wall time of the lease but not in a specific deadline.

[1] http://aws.amazon.com/ec2/spot-instances

Suspendable leases are flexible in start time and they can be scheduled at any time after their ready time. In the case of preemption, these leases should be rescheduled to find another free time-slot for the remainder of their execution. The overhead time of preempting a Suspendable lease is sum of the time needed to suspend a VM, reschedule the lease, and resume it later. Suspendable leases are suitable for Bag-of-task (BOT) and Parameter Sweep type of applications (Buyya et al. 2005).

- *Restartable* (Redirectable): In this case the preempted lease can be canceled and restarted in another Grid later on. In InterGrid IGGs can redirect requests to other Grids through peering adjustments (De Assunção et al. 2008). Restartable requests can be either best-effort or deadline-constraint. In the former case, the wall time of the request and in the latter both wall time and deadline of grid request is gauranteed. However, we do not consider this type of grid leases in our preemption policies and leave this choice for the future work.

- *Deadline Constraint-Migratable*: These leases guarantee both the wall time and deadline of the lease. However, there is no guarantee that they will be run on a specific resource(s). In other words, there is always a chance for the lease to be preempted but it will be resumed and finished before its deadline, either on the same resource or on another resource. Nonetheless, migrating VMs involves VM transferring overhead. One solution to mitigate this overhead is migrating to another RP inside the same Grid of InterGrid which has a high bandwidth connection. Multiple reservation is also a conceivable strategy to serve such kind of leases (Sotomayor et al. 2008). We leave the details of migration issues involved as a future work. Migratable requests are needed by steerable applications (Costanzo et al. 2009). In these applications, which are already implemented in InterGrid, the workload can be migrated to more powerful sites to meet user constraints such as deadline (Costanzo et al. 2009).

- *Deadline Constraint-Non-Preemptable*: The leases associated with such requests cannot be preempted at all. These leases guarantee both deadline and wall time without being preempted during the lease. This type of lease is useful for critical tasks in workflows where some tasks have to start and finish at exact times to prevent delaying the execution of the workflow (Kwok & Ahmad 1996).

We assume that local requests are all deadline constraint non-preemptable. However, grid users can send all request types mentioned above. Different lease types correspond to different prices. Thus, users are motivated to associate their requests to different request types. Unarguably, the more flexible request type the less expensive the lease. However, we leave the market-oriented implications of the lease-based scheduling as a future work.

### 4.2 Preemption Policies

In this section we discuss policies for choosing the best candidate set for preemption.

As mentioned earlier, the scheduler in this system faces with two types of requests: grid requests and local requests.

If the scheduler receives a non-preemptable or Migratable grid request, the scheduler must determine whether there are adequate free resources available for the requested wall time from the requested start time. If the request is found not to be possible, it will be rejected. Nonetheless, there is more flexibility for Suspendable and Cancelable requests. In fact, since such requests are not restricted to any specific deadline, the scheduler tries to find a vacant place for the requested wall time in any available time-slot.

If the scheduler cannot find any vacant resource for a local request, then the scheduler tries to make room for the local request by preempting the Suspendable, Cancelable, and/or Migratable leases that coincide with the local request's needed interval. Thus, leases that their preemption makes enough space for the local request are selected and form all candidate sets. Each candidate set contains a set of leases that their preemption makes enough space for an incoming local request. However, if resources are occupied by non-preemptable grid leases or leases from other local requests, then the local request inevitably gets rejected. Preemption policy in this situation determines the proper candidate set for preemption.

Choosing different candidate sets affects the number of VMs to be preempted, which turns out to present different time overheads. On the other hand, selecting different candidate sets leads to a different number of leases to be preempted, which adds more waiting time and, consequently, more grid user dissatisfaction.

We propose three preemption policies that result in different candidate sets to be preempted. They lead to a different amount of time overhead and a different number of leases to get preempted. One policy focuses on system centric criteria by trying to increase resource utilization. The second policy focuses on user centric criteria and tries to preempt fewer leases to make more user satisfaction. The third policy makes a trade-off between resource utilization and user satisfaction.

#### 4.2.1 Minimum Overhead Policy (MOV)

As a system centric policy, this policy aims at maximizing resource utilization. Therefore, this policy tries to minimize the time overhead imposed to the underlying system by preempting a candidate set that leads to the minimum overhead. For this purpose the total overhead imposed to the system by each candidate set is calculated and a set with minimum overhead is selected. According to the notation we defined in Section 2, MOV policy can be presented based on Equation 3.

$$MOV(A) = \min_{m=0}^{S-1}\{h(C_m)\} \tag{3}$$

The overhead time imposed by each candidate set varies based on the type of leases involved in that candidate set (Sotomayor et al. 2009). For *Cancelable* leases the overhead is the time needed to terminate the lease and shut down its VM(s). This time is usually much lower than the time needed for Suspending or Migrating leases (Sotomayor et al. 2009).

The overhead imposed by preempting a *Suspendable* lease includes the time needed to suspend and resume VM(s) plus a time for re-scheduling the remaining time of the lease. The estimation of these times has been carried out by Sotomayor et al. (Sotomayor et al. 2009). We use the same method to work out the overhead time imposed by preempting Suspendable leases. Therefore, the overhead of suspension is $t_s = \frac{mem}{s}$ and resumption time is $t_r = \frac{mem}{r}$. Where $mem$ is the amount of VM memory, $s$ is the rate of suspending megabytes of VM memory per second,

and $r$ is the rate of re-allocating megabytes of VM memory per second (Sotomayor et al. 2009). Preempting a *Migratable* lease enforces VM(s) transferring overhead in addition to the overheads mentioned for Suspendable leases (Zhao & Figueiredo 2007).

Since we consider a global file system, if there are several VMs in a lease (i.e., $K$), then the preemption overhead time is multiplied by $K$ (Sotomayor et al. 2009).

#### 4.2.2 Minimum Leases Involved Policy (MLIP)

Users do not like that their leases get affected by preemption. In fact, preempting leases makes longer waiting times for Suspendable and Migratable leases to get completed. In the case of Cancelable leases, preempting the lease results in terminating the lease. Therefore, as a user centric policy, MLIP tries to satisfy more users by preempting fewer leases.

In this policy a candidate set that contains minimum number of leases is selected from all the candidate sets. MLIP disregards the type of leases involved in a candidate set during decision making. Based on the notation introduced in Section 2, MLIP can be presented according to Equation 4.

$$MLIP(A) = \min_{m=0}^{S-1}\{|C_m|\} \tag{4}$$

where $|C_m|$ gives the number of leases involved (cardinality) in each candidate set $C_m$.

#### 4.2.3 Minimum Overhead Minimum Lease Policy (MOML)

The two proposed policies mentioned earlier aim to either improve resource utilization (as a system centric criterion) or minimize the number of preempted leases (as a user centric criteria). However, in MOML we propose an approach that can fulfill both system and user centric criteria at the same time. This policy is depicted in Figure 2 and elaborated in Algorithm 1. In fact, MOML is a balance between MOV, which minimizes the imposed overhead, and MLIP, which attempts to minimize the number of requests affected by preemption.

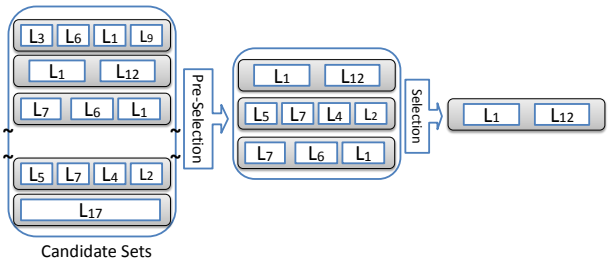


Figure 2: Pre-selection and final selection steps of MOML policy.

According to Figure 2 and Algorithm 1, in MOML the selection of a candidate set is carried out in two steps. In the first step (pre-selection) all candidate sets which have a total overhead less than a certain threshold ($\alpha$) are pre-selected for the second step (lines 5 to 8 in Algorithm 1). The first step increases the tolerance of acceptable overhead in comparing with MOV. In the second step, to have fewer leases affected, a candidate set that contains minimum number of leases is selected(lines 9 to 11 in Algorithm 1).

Selecting a proper value for $\alpha$ determines the behaviour of MOML policy. More specifically, if the $\alpha \to \infty$, then MOML behaves the same as MLIP. On the other hand, if $\alpha \to 0$, then MOML behaves the same as MOV. Thus, to keep the trade-off between MOV and MLIP, we consider $\alpha$ 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 step for having a minimum number of leases.

**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;
```

## 5 Performance Evaluation

In this section we discuss different performance metrics considered, the scenario in which the experiments are carried out, and experimental results obtained from simulation.

### 5.1 Performance Metrics

Introducing different types of leases along with preemption policy are expected to affect different parameters, which are described over the next subsections.

#### 5.1.1 Local and Grid Request Rejection Rate

The initial goal of this research is serving more local requests by preempting resources from grid leases. Therefore, as a result of our research, it is interesting for us to find out how efficient these preemption policies are in terms of serving more local requests.

The "local request rejection rate" is the fraction of local requests which are rejected, possibly because of allocating resources to non-preemptable grid requests or other local requests.

Additionally, we are interested to see if decreasing local request rejection rate comes with the cost of rejecting more grid requests. Grid Request Rejection Rate describes this metric and shows the percentage of grid requests that are rejected. The ideal case is that local request rejection rate is reduced without increasing the grid request rejection rate.

#### 5.1.2 Resource Utilization

Time overhead is a side-effect of preempting leases that results in resource under-utilization. Therefore, we are interested to see how different preemption policies affect the resource utilization.

Resource utilization is defined according to the Equation 5.

$$Utilization = \frac{computationTime}{totalTime} * 100 \tag{5}$$

Table 1: Detailed specifications of the generated workloads. $|BE|$ stands for the number of best effort grid requests, $|DC|$ stands for the number of deadline constraint grid requests, and $|Local|$ stands for the number of local request.

| Modified Parameter | Distribution | Other Constant Parameters |
|---|---|---|
| Average No. VMs | Two-stage uniform | $\|BE\|$ =1000, $\|DC\|$ =1000, $\|Local\|$ =1000 |
| No. BE Requests | Uniform | Average No. VMs=4, $\|DC\| = 2000 - \|BE\|$, $\|Local\|$ =1000 |
| No. DC Requests | Uniform | Average No. VMs=4, $\|BE\| = 2000 - \|DC\|$, $\|Local\|$ =330 |
| No. Local Requests | Uniform | Average No. VMs=4, $\|BE\| = \|DC\| = (totalRequest - local)/2$ |

Where:

$$computationTime = \sum_{i=1}^{|\lambda|} v(l_i) \cdot d(l_i) \qquad (6)$$

Where $|\lambda|$ is the number of leases, $v(l_i)$ is the number of VMs in lease $l_i$, $d(l_i)$ is the wall time of lease $l_i$.

#### 5.1.3 Number of Lease Preemption

Preempting grid leases has different impacts on different 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. For instance, owners of Migratable leases expect to get their accepted and meet the deadline while owners of Suspendable leases like to have short waiting times. Nonetheless, in all types of leases grid users suffer from lease preemption. Therefore, to have a generic metric to measure the user satisfaction, we are interested to see the total number of preemptions resulted by different policies.

### 5.2 Experimental Setup

We used a discrete event simulator to evaluate the performance of the preemption policies. These preemption policies are implemented in the context of InterGrid. In the experiments conducted, Lublin99 (Lublin & Feitelson 2001) has been configured to generate 3000 parallel jobs.

Lublin99 is the workload model based on the San Diego Super Computer (SDSC) Blue Horizon machine. Job traces collected from this supercomputer are publicly available and have been studied extensively in the past.

We consider a cluster with 32 nodes as an RP. We assume all nodes of the RP as single core with one VM. The maximum number of VM(s) in generated requests is also 32.

We consider each VM of 1024 MB and a 100 Mbps network bandwidth. Hence, according to Section 4.2.1, in our experiments, suspension time ($t_s$) and resumption time ($t_r$) are 161 and 126 seconds respectively. The time overhead for migrating a VM with similar configuration is 160 seconds (Zhao & Figueiredo 2007).

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

- different number of VMs needed.
- different number of best effort grid requests (including Cancelable and Suspendable).
- different number of deadline constraint grid requests (including Migratable and Non-Preemptable).
- different number of local requests.

Each experiment is carried out on each of these workloads separately. To generate these workloads, we modify parameters of Lublin99's model. The way these workloads are generated and other detailed specifications of these workloads are described in Table 1.

### 5.3 Experimental Results

#### 5.3.1 Local and Grid Request Rejection Rate

The primary goal of this paper is to show the impact of preempting grid leases to allocated resources to local requests. In Table 2, the mean difference of decrease in local requests rejection rate is reported along with a 95% confidence interval of the difference. We report the difference between rejection rate in two situations; First, when no preemption policy is used and second, when MOML is used as a preemption policy. Since all proposed preemption policies resulted in similar local and grid rejection rate we have just reported the result for MOML. We use a T-test to work out the mean difference between these two policies. To perform the T-test we have ensured that the distribution of difference is normal.

According to Table 2, local request rejection rate has statistically and practically significantly decreased by applying preemption in all cases. More importantly, this reduction in local request rejection rate has not been with the cost of rejecting more grid requests. Based on Table 2, it can be noted that in all experiments grid request rejection rate does not change significantly.

Based on this experiment, the maximum decrease in local request rejection rate occurs when the percentage of best effort grid requests is higher (second row in Table 2). In this circumstance, more local requests can be accommodated by preempting these best effort leases.

#### 5.3.2 Resource Utilization

In this experiment we measure the resources utilization when different preemption policies are applied. As illustrated in all sub-figures of Figure 3, we explore the impact of altering workload parameters pointed out in Table 1 on resource utilization when different preemption policies are applied.

This experiment indicates that resource utilization increases almost linearly by increasing the average number of VMs in requests (Figure 3(a)). Although preempting best effort leases make some overhead, we can see in Figure 3(b) that increasing the number of best effort requests improves resource utilization;

Table 2: Mean difference and 95% confidence interval (CI) of decrease in local requests rejection rate and grid requests rejection rate as a result of applying preempting leases in an RP of InterGrid.

| **Modified Parameter** | **Mean Decrease in Local Requests Rejection Rate** | **CI of Decrease in Local Requests Rejection Rate** | **Change in Grid Requests Rejection Rate** |
|---|---|---|---|
| Average No. of VMs | 55.1% | (47.2,62.9), P-Value<0.001 | Equal |
| Percentage of BE Grid Requests | **72.0**% | (51.1,92.8), P-Value=0.001 | Not statistically significant, P-Value=0.6 |
| Percentage of DC Grid Requests | 54.3% | (35.0,73.7), P-Value=0.001 | Not statistically significant, P-Value=0.3 |
| Percentage of Local Requests | 58.2% | (40.3,75.9), P-Value<0.001 | Not statistically significant, P-Value=0.6 |

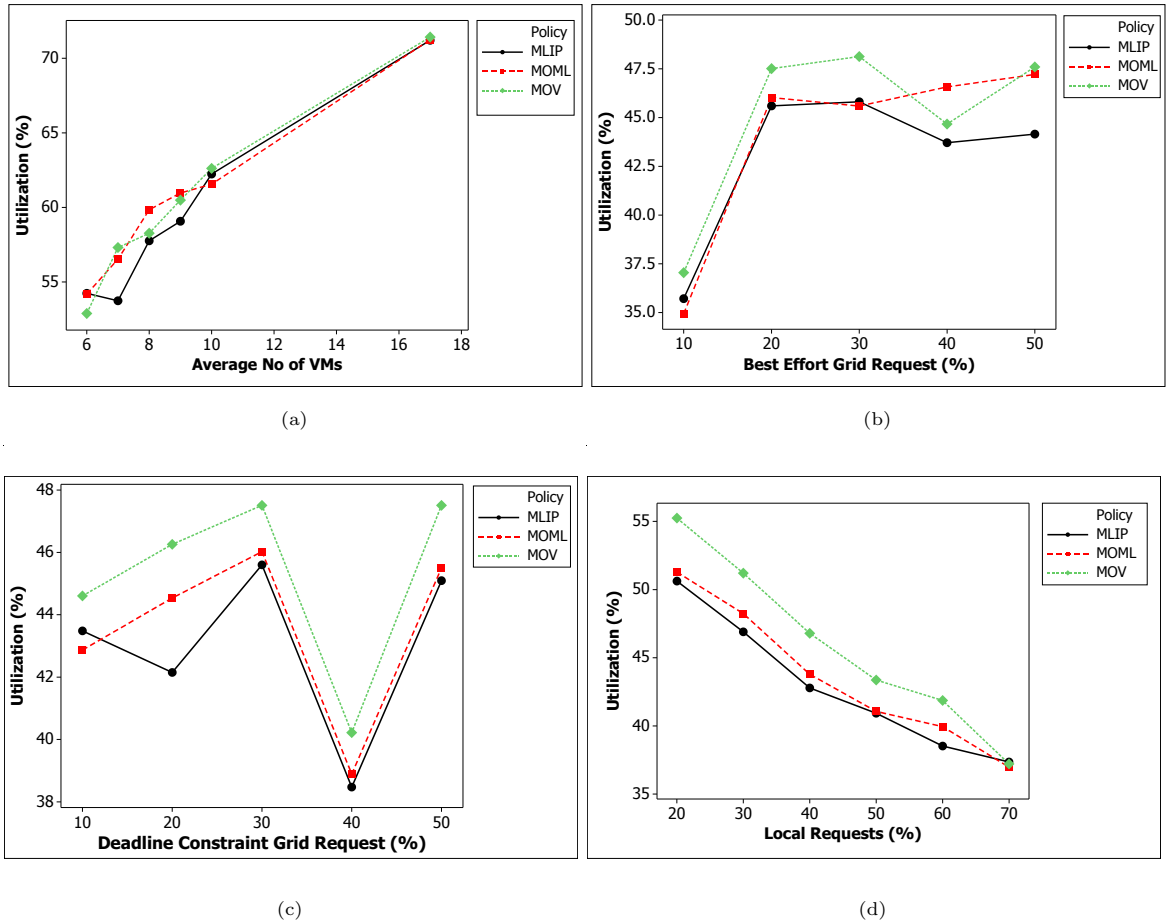


Figure 3: Resource utilization results from different policies. This experiment was carried out by modifying (a) the average number of VMs, (b) the percentage of best effort grid requests, (c) the percentage of deadline constraint grid requests, and (d) the number of local requests.

however, after a certain percent (best effort>20%) resource utilization does not fluctuate significantly in different policies. The reason is that, allocating other (unused) time-slots to the preempted leases result in less resource fragmentation and therefore resource utilization does not decrease by increasing the percentage of the grids best effort requests.

In Figure 3(c), we can see that resource utilization increases by increasing the percentage of deadline constraint requests in all policies. However, there is a sharp decrease (from around 45% down to 38%) when 40% of requests are deadline constraint. In fact, in this point there are many best effort and deadline constraint requests in the system at the same time. Hence, more preemption occurs and subsequently more overhead is imposed to the system. We can conclude that the system would result in minimum utilization when 40% of requests are deadline constraint and the rest are best effort.

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 3(d), by increasing the number of local requests, resource utilization decreases almost linearly in all policies.

In all sub-figures of Figure 3 it can be observed that MOV result in better utilization comparing with the other policies. However, in a few points (e.g., 40 in Figure 3(b)), MOV has slightly less utilization than MOML. This happens mainly because of the resource fragments that occur at scheduling time which leads to resource under-utilization. Sub-figures of Figure 3 also demonstrates that resource utilization MOML lies between MLIP and MOV.

### 5.3.3 Number of Lease Preemptions

In this experiment we investigate the number of grid leases that get preempted by applying different preemption policies.

From Figure 4(a) we can infer that, in general, larger leases (i.e. 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

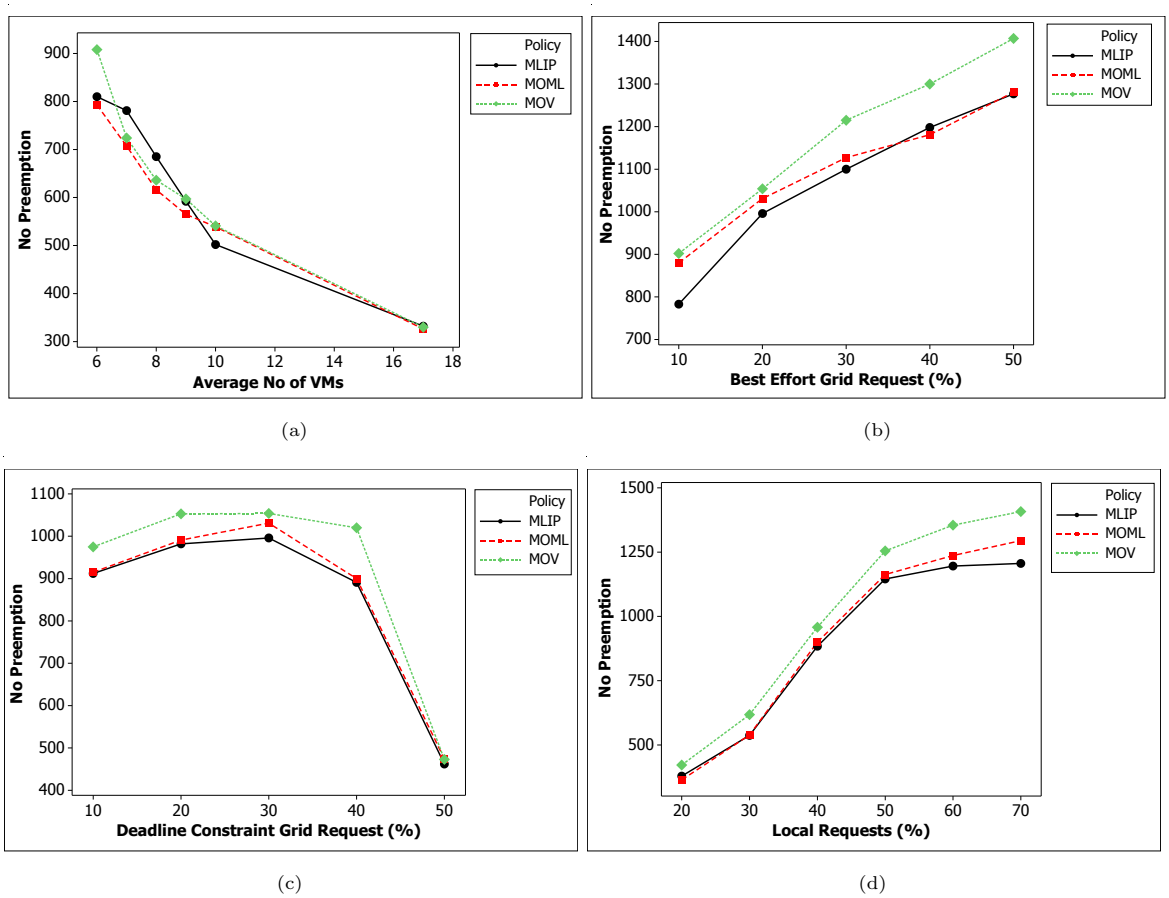


Figure 4: Number of lease preemption resulted from different policies by changing (a) the average number of VMs, (b) percentage of best effort grid requests, (c) percentage of deadline constraint grid requests, and (d) the number of local requests.

room for incoming local requests.

Figure 4(b) shows that by increasing the number of best effort grid requests the number of preemptions increases almost linearly. For a lower percentage of best effort grid requests (percentage best effort<30%), MOML behaves similar to MOV, however, after that point MOML approaches to MLIP. The reason is that when the number of Suspendable leases is high, the likelihood of having a candidate set with minimum number of leases and not large overall overhead is high. Thus, MOML and MLIP approach each other.

In Figure 4(c), the number of preemptions does not vary significantly when the percentage of deadline constraint grid requests is increased. However, when the percentage of deadline constraint grid requests exceeds 40% the number of preemptions drops sharply (from around 1000 to 500). In fact, best effort grid requests result in an excessive number of preemptions and therefore, when the proportion of deadline constraint requests increases, the number of preemptions decrease significantly.

Figure 4(d) demonstrates that the number of preemptions increases by increasing the number of local requests in all policies almost linearly.

As illustrated in all sub-figures of Figure 4, most of the time MLIP results in a minimum number of preemptions and MOML operates between MLIP and MOV. The only exceptions are in points 7, 8 of Figure 4(a) where MLIP makes more preemptions rather than MOV and MOML. We believe that on these points MLIP has preempted some leases which had short wall times. Therefore, after preemption they are allocated in a close time-slot and again these leases are preempted by MLIP.

## 6 Conclusion and Future Work

In this research we explored how local requests of an RP can get access to occupied resources in InterGrid. For this purpose we leveraged preempting grid leases in favor of local requests. We proposed different types of leases plus different policies to decide which lease(s) are better choices for preemption. More specifically, we investigated three policies for lease preemption. MOV as a policy that improves system utilization, MLIP that results in fewer preemption and increasing user satisfaction, and MOML which makes a trade-off between resource utilization and user satisfaction.

We observed that preempting leases substantially decrease the rejection of local requests (up to 72% with 95% CI:(51.1,92.8)) without increasing grid requests rejection rate. These results are the same for all proposed preemption policies. We also noticed that MOV performs better in terms of resource utilization in comparing with other policies. On the other hand, MLIP is a better policy in the sense that it preempts fewer leases and therefore causes more user satisfaction. MOML is a policy which satisfies both resource utilization and the user at the same time.

Although the problem we are investigating in this paper is in InterGrid context, it could be also applied to other lease-based Grid/Cloud resource providers where requests with higher priority (such as local or organisational requests) coexist with other requests.

In the future, we plan to extend the current work by considering circumstances where there is a dependency between leases. Furthermore, we are interested in scenarios where local requests are also from different types (e.g. local Suspendable and local Migratable).

## References


Buyya, R., Murshed, M. M., Abramson, D. & Venugopal, S. (2005), 'Scheduling parameter sweep applications on global grids: a deadline and budget constrained cost-time optimization algorithm', *Softw., Pract. Exper.* **35**(5), 491–512.

Chase, J. S., Irwin, D. E., Grit, L. E., Moore, J. D. & Sprenkle, S. E. (2003), Dynamic virtual clusters in a grid site manager, *in* 'Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing', Washington, DC, USA, pp. 90–98.

Costanzo, A. d., Jin, C., Varela, C. A. & Buyya, R. (2009), Enabling computational steering with an asynchronous-iterative computation framework, *in* 'E-SCIENCE '09: Proceedings of the 2009 Fifth IEEE International Conference on e-Science', IEEE Computer Society, Washington, DC, USA, pp. 255–262.

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

Grit, L., Ramakrishnan, L. & Chase, J. (2007), On the duality of jobs and leases, Technical Report CS-2007-00, Duke University, Department of Computer Science.

Isard, M., Prabhakaran, V., Currey, J., Wieder, U., Talwar, K. & Goldberg, A. (2009), Quincy: fair scheduling for distributed computing clusters, *in* 'Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles (SOSP)', ACM, pp. 261–276.

Kettimuthu, R., Subramani, V., Srinivasan, S., Gopalsamy, T., Panda, D. K. & Sadayappan, P. (2005), 'Selective preemption strategies for parallel job scheduling', *IJHPCN* **3**(2/3), 122–152.

Kwok, Y.-K. & Ahmad, I. (1996), 'Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors', *IEEE Trans. Parallel Distrib. Syst.* **7**(5), 506–521.

Lublin, U. & Feitelson, D. G. (2001), 'The workload on parallel supercomputers: Modeling the characteristics of rigid jobs', *Journal of Parallel and Distributed Computing* **63**, 1105–1122.

Snell, Q., Clement, M. J. & Jackson, D. B. (2002), Preemption based backfill, *in* 'Job Scheduling Strategies for Parallel Processing (JSSPP)', Springer, pp. 24–37.

Sotomayor, B., Keahey, K. & Foster, I. (2008), Combining batch execution and leasing using virtual machines, *in* 'Proceedings of the 17th International Symposium on High Performance Distributed Computing', ACM, New York, NY, USA, pp. 87–96.

Sotomayor, B., Montero, R. S., Llorente, I. M. & Foster, I. (2009), Resource leasing and the art of suspending virtual machines, *in* 'Proceedings of the 11th IEEE International Conference on High Performance Computing and Communications', Washington, DC, USA, pp. 59–68.

Walters, J., Bantwal, B. & Chaudhary, V. (2008), 'Enabling interactive jobs in virtualized data centers', *Cloud Computing and Applications* pp. 21–26.

Zhao, M. & Figueiredo, R. (2007), 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', ACM, pp. 5–11.