J. Parallel Distrib. Comput. 72 (2012) 231–245

Contents lists available at SciVerse ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

QoS and preemption aware scheduling in federated and virtualized Grid computing environments

Mohsen Amini Salehi, Bahman Javadi, Rajkumar Buyya *

Cloud Computing and Distributed Systems (CLOUDS) Laboratory, Department of Computing and Information Systems, The University of Melbourne, Parkville, Melbourne, VIC, 3010, Australia

ARTICLE INFO

Article history:
Received 14 April 2011
Received in revised form
10 October 2011
Accepted 24 October 2011
Available online 10 November 2011

Keywords:
Federated Grid
Virtual Machine (VM)
Preemption
Queuing model
Scheduling
Quality of Service (QoS)

ABSTRACT

Resource provisioning is one of the challenges in federated Grid environments. In these environments each Grid serves requests from external users along with local users. Recently, this resource provisioning is performed in the form of Virtual Machines (VMs). The problem arises when there are insufficient resources for local users to be served. The problem gets complicated further when external requests have different QoS requirements. Serving local users could be solved by preempting VMs from external users which impose overheads on the system. Therefore, the question is how the number of VM preemptions in a Grid can be minimized. Additionally, how we can decrease the likelihood of preemption for requests with more QoS requirements. We propose a scheduling policy in InterGrid, as a federated Grid, which reduces the number of VM preemptions and dispatches external requests in a way that fewer requests with QoS constraints get affected by preemption. Extensive simulation results indicate that the number of VM preemptions is decreased at least by 60%, particularly, for requests with more QoS requirements.

1. Introduction

Resource provisioning for user applications is one of the main challenges and research areas in federated Grid environments. Federated Grids, such as InterGrid, enable sharing, selection, and aggregation of resources across several Grids, which are connected through high bandwidth network connections. Nowadays, heavy computational requirements, mostly from scientific communities, are supplied by these federated environments such as PlanetLab [8]. 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 *lease* abstraction [38].

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 [12]. In each constituent Grid, the provisioning rights over several clusters inside the Grid are delegated to the InterGrid Gateway (IGG). IGGs coordinate resource allocation for requests coming from other Grids (external users) through predefined contracts between Grids [11]. 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 Fig. 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 a larger amount of shared resources. Typically, local requests have priority over external requests in each cluster [6]. 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.

In our previous research [33], we demonstrated how preemption of external requests in favor of local requests can help serving more local requests. However, the side-effects of preemption are twofold:

- From the system owner perspective, preempting VMs imposes a notable overhead to the underlying system and degrades resource utilization [38].
- From the external user perspective, preemption increases the response time of the external requests.

As a result, both the resource owner (who prefers to increase resource utilization) and external users (who are interested in

* Corresponding author.
E-mail addresses: mohsena@csse.unimelb.edu.au (M. Amini Salehi), bahmanj@csse.unimelb.edu.au (B. Javadi), raj@csse.unimelb.edu.au (R. Buyya).

0743-7315/$ – see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2011.10.008

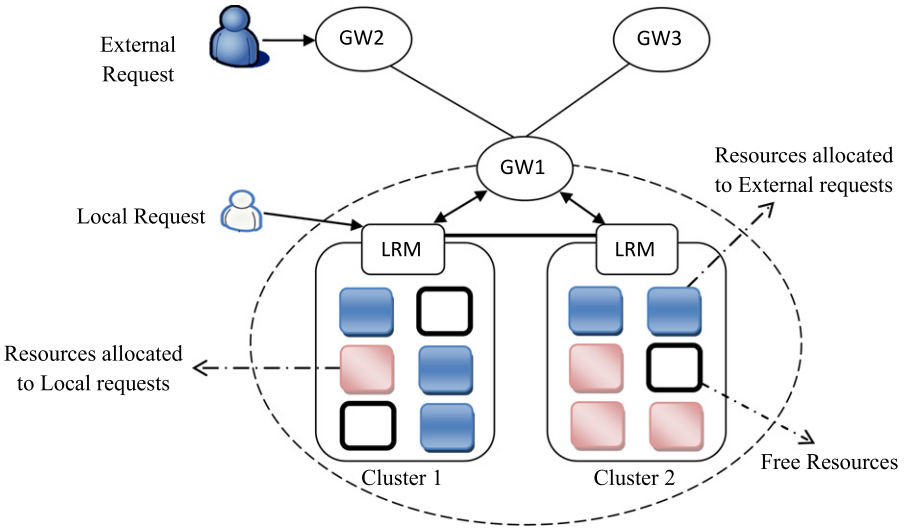

Fig. 1. A scenario that shows the contention between local and external requests in a federated Grid environment (InterGrid) [33].

shorter response time) benefit from fewer VM preemptions in the system. We believe that with the extensive current trend in applying VMs in distributed systems, and considering preemption as an outstanding feature of VMs, it is crucial to investigate policies that minimize these side-effects. Therefore, one problem we are dealing with in this research is how to decrease the number of VM preemptions that take place in a virtualized Grid environment.

The problem gets complicated further when external requests have different levels of Quality of Service (QoS) requirements (also termed different request types in this paper). For instance, some external requests can have deadlines whereas others do not. Preemption affects the QoS constraints of such requests. This implies that some external requests are *more valuable* than others and, therefore, more precedence should be given to valuable requests by reducing the chance of preemption of these requests.

To address these problems, in this paper, we propose a QoS and preemption-aware scheduling policy for a virtualized Grid which contributes resources to a federated Grid. This scheduling policy comprises of two parts.

The first part, called workload allocation policy, determines the fraction of external requests that should be allocated to each cluster in a way that minimizes the number of VM preemptions. The proposed policy is based on the stochastic analysis of routing in parallel, non-observable queues. Moreover, this policy is knowledge-free (i.e. it is not dependent on the availability information of the clusters). Thus, this policy does not impose any overhead on the system. However, it does not decide the cluster that each single external request should be dispatched upon arrival. In other words, dispatching of the external requests to clusters is random.

Therefore, in the second part, called dispatch policy, we propose a policy to find out the cluster to which each request should be allocated to. The dispatch policy has the awareness of request types and aims to minimize the likelihood of preempting valuable requests. This is performed by working out a deterministic sequence for dispatching external requests. In summary, our paper makes the following contributions:

- Providing an analytical queuing model for a Grid, based on the routing in parallel non-observable queues.
- Adapting the proposed analytical model to a preemption-aware workload allocation policy.
- Proposing a deterministic dispatch policy to give more priority to more valuable users and meet their QoS requirements.
- Evaluating the proposed policies under realistic workload models and considering performance metrics such as number of VM preemptions, utilization, and average weighted response time.

We utilize InterGrid [10], which is a virtualized federated Grid environment, as the context of our work. In the next section, InterGrid structure is discussed in detail. The rest of this paper is organized as follows: In Section 2, an overview of the InterGrid environment is provided. The proposed analytical queuing model is described in Section 3 which is followed by the preemption-aware scheduling policy in Section 4. Performance evaluation of the proposed policy is reported in Section 5. Then, in Section 6 related research works are introduced. Finally, conclusion and future works are provided in Section 7.

2. InterGrid environment

In this section, we provide a brief overview on InterGrid architecture and implementation. Interested readers could refer to [12] for more details.

2.1. Architecture

In InterGrid each Grid has predefined peering arrangements with other Grids, which are managed by IGGs and through which IGGs coordinate the adoption of InterGrid's resources. An IGG is aware of the peering terms between Grids, selects suitable Grids that can provide the required resources, and replies to requests from other IGGs.

The Local Resource Manager (LRM)[1] is the resource manager in each cluster which provisions resources for local and external (Grid) requests. Resource provisioning in clusters of InterGrid 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 [38]. Virtual Machine (VM) technology is used in InterGrid to implement lease-based resource provisioning [39]. InterGrid creates one *lease* for each user *request* (in this paper we apply these two terms interchangeably).

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

In InterGrid each request is contiguous and has to be served within resources of a single cluster. Each request has the following characteristics:

- Type of the request.
- Number of VMs.
- Duration for the request.
- Deadline for the request (optional).

We consider several types of external requests in InterGrid which correspond to different QoS levels. These external requests are broadly classified as Best-Effort (BE) and Deadline-Constraint (DC) requests. BE requests, are scheduled in the first available time-slot. In the case that there are not enough resources to start, BE external requests can be preempted in favor of local requests. DC requests are rejected if there is not enough resources for them to start. Based on this categorization, we classify external requests as follows:

- *BE-Cancelable*: these leases neither guarantee the deadline nor the duration of the lease. These leases can be started at any time after their ready time. At the time of preemption, if the cluster is overloaded, Cancelable leases are terminated. Preempting Cancelable leases impose very little overhead time. This overhead pertains to the required time for terminating VMs allocated to the lease. Cancelable leases are applicable for map-reduce requests [20]. Spot instances in Amazon EC2[2] are another example of Cancelable leases.
- *BE-Suspendable*: these leases ensure the duration of the lease but not in a particular deadline. Similar to Cancelable leases, these leases can be scheduled at any time after their ready time. In the case of preemption, these leases are re-scheduled to find another time-slot for the rest of execution. The overhead time of preemption in this case is the amount of time required to suspend VMs, reschedule the lease, and resume it later [39]. Suspendable leases are appropriate for Parameter Sweep and Bag-of-Task applications [26].
- *DC-Migratable*: Duration and the deadline have to be guaranteed for these type of leases. A Migratable lease can be preempted but it has to be resumed and finished before its deadline, either on the same resource or on another resource. Migrating VMs entail VM transferring overhead. One solution to alleviate this overhead is migrating the VM to another cluster inside the same Grid of InterGrid which has a high bandwidth connection. We leave the details of VM migration issues as a future work. Migratable requests are applicable where the job has a loose deadline or requires more powerful resources.
- *DC-Non-Preemptable*: These leases guarantee both deadline and duration without getting preempted during the lease. This type of lease is useful for jobs with very tight deadline. For example, critical tasks in work-flows where some tasks have to be started and finished at exact times to prevent delaying the execution of the work-flow [23,27].

We assume that local requests are all DC-Non-preemptive whereas external users can send all request types mentioned earlier. Hence, there is a mixture of different requests entering to each Grid. The case that local requests have different types is also interesting. However, in this work we consider several types of external requests and leave different types of local requests as a future work.

Different lease types incur different costs for the user. Thus, users are encouraged to request various lease types. Unarguably, the less flexible the request type is, the more expensive the lease will be. We consider the more expensive leases (i.e. DC) as more valuable ones and take this criteria into account at the time of scheduling.

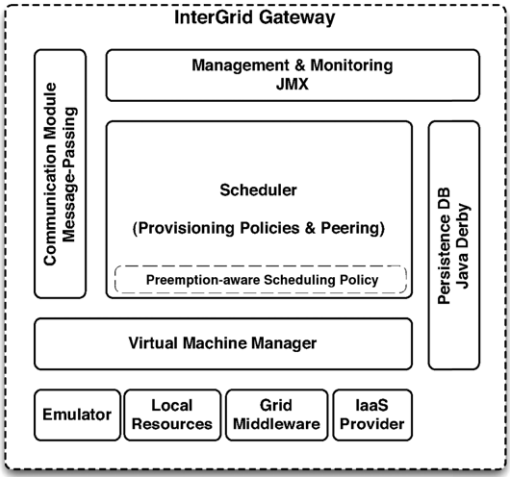

Fig. 2. InterGrid Gateway components.

2.2. Implementation

The core part of InterGrid, IGG, has been implemented in Java. A layered view of its components is depicted in Fig. 2. The core component of IGG is the *Scheduler*, which implements provisioning policies and peering with other IGGs. The scheduler performs creation, starting, and stopping VMs through the Virtual Machine Manager (VMM). VMM implementation is generic, so different LRMs can interact with it. Currently, it is possible for a VMM to connect to OpenNebula [13], or Eucalyptus [30] to manage local resources. In addition, two interfaces to connect to a Grid middleware (i.e., Grid'5000) and a Cloud IaaS provider (i.e., Amazon EC2[3]) have been developed. Moreover, an emulated LRM for testing and debugging has been implemented for the VMM.

The persistence database is used for storing information of IGG such as VM templates and peering arrangements. The Management and Monitoring provide command-line tools to configure and manage IGG. The Communication Module provides an asynchronous message-passing mechanism between IGGs, which makes IGGs loosely coupled and fault-tolerant.

3. Analytical queuing model

In this section, we describe the analytical modeling of preemption in a virtualized Grid environment based on routing in parallel queues. This section is followed by our proposed scheduling policy in IGG 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 external requests) is depicted in Fig. 3. There are N clusters where 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 external requests which are sent by the gateway with arrival rate $\hat{\Lambda}_j$. The gateway receives external requests from other peer gateways [12] ($G_1, \ldots, G_{peer}$ in Fig. 3). Therefore, external request arrival rate to the gateway is $\Lambda = \bar{\Lambda}_1 + \bar{\Lambda}_2 + \cdots + \bar{\Lambda}_{peer}$ where *peer* indicates the number of gateways that can potentially send external requests to the gateway. Submitted local requests to cluster j must be executed on cluster j unless the requested resources are occupied by another local request or a Non-preemptive external request (see Section 2).

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

[3] http://aws.amazon.com/ec2.

Table 1
Description of symbols used in the queuing 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 external requests to cluster j
$\hat{\Lambda}_j$	Arrival rate of external requests to cluster j after load distribution
Λ	$= \sum_{i=1}^{peer} \bar{\Lambda}_i = \sum_{j=1}^{N} \hat{\Lambda}_j$
θ_j	Average service time of a external request on cluster j
ω_j	Second moment of external requests service time on cluster j
γ_j	$= \theta_j \cdot \hat{\Lambda}_j$
λ_j	Arrival rate of local requests to cluster j
κ_j	Arrival rate of local requests plus external 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	$= \frac{\lambda_j}{\kappa_j}\omega_j + \frac{\lambda_j}{\kappa_j}\mu_j$
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 external requests
T_j	Average response time of external requests on cluster j
$\bar{v}_j$	Average number of VMs required by external requests
$\bar{d}_j$	Average duration of external requests
s_{ij}	Processing speed (MIPS) of processing element i in cluster j

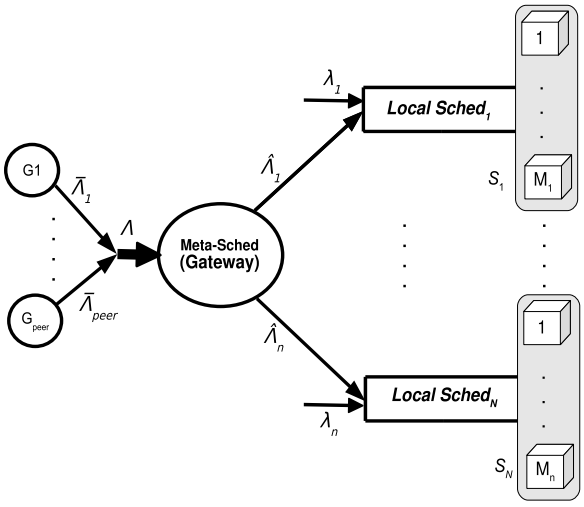

Fig. 3. Queuing model for resource provisioning in a Grid with N clusters.

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, an external 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 external 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 external requests (Λ) amongst the clusters. In this situation if we consider each cluster as a single queue and the gateway as a meta-scheduler that redirects each incoming external request to one of the clusters, then the problem of scheduling external requests in the gateway (IGG) can be considered as a routing problem in distributed parallel queues [2].

Considering these situation, the goal of the scheduling in the IGG is to schedule the external 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:

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

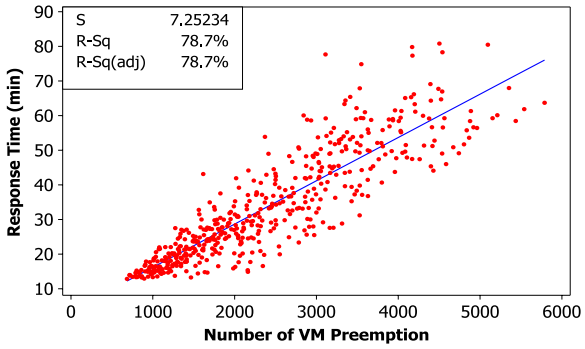

Fig. 4. Regression between the number of VMs preempted and response time of external requests.

To the best of our knowledge, there is no scheduling policy for such an environment with the goal of minimizing the number of VM preemptions. However, several research works have been undertaken in similar circumstances to minimize the average response time of external requests.

Some initial experiments as well as results of our previous research [33] intuitively imply that there is an association between response time and number of VM preemptions in the Grid. The regression analysis with least squares method (depicted in Fig. 4 and shown in Eq. (2)) demonstrates the positive correlation between the two factors. In Eq. (2), R and η indicate the response time of external requests and number of VM preemptions, respectively

$$R = 3.09 + 0.012\eta. \tag{2}$$

Therefore, we expect that minimizing average response time have similar impact on the overall number of VM preemptions. Simulation results, which are discussed in Section 5.3, also confirm the correlation of response time and number of VM preemptions in the system. Details of the analysis are discussed over the next paragraphs.

For this purpose, we extend the approach developed by Li [24], which has been done within a cluster, for circumstances that there is a Grid system in which some external requests are more valuable than others (i.e. different levels of QoS).

Thus, we can define a new objective function that aims at minimizing the average response time of external requests (Eq. (3)):

$$T = \frac{1}{\Lambda} \sum_{j=1}^{N} \hat{\Lambda}_j \cdot T_j. \tag{3}$$

Given the $M/G/1$ queue for each cluster, and also preempting external requests in favor of local requests, then the response time of external requests in cluster j (T_j) is worked out based on Eq. (4) [22]

$$T_j = \frac{1}{1-\rho_j}\left(\theta_j + \frac{\kappa_j m_j}{2(1-u_j)}\right). \tag{4}$$

The constraint for Eq. (3) is:

$$\sum_{j=1}^{N} \hat{\Lambda}_j - \Lambda = 0. \tag{5}$$

The Lagrange multiplier method is applied to minimize Eq. (3). We consider Eq. (3) as $f(\hat{\Lambda}_j)$, Eq. (5) as $g(\hat{\Lambda}_j) - c$, and z as the Lagrange multiplier. Then, the Lagrange function is defined as follows:

$$\begin{aligned} h(\hat{\Lambda}_j, z) &= f(\hat{\Lambda}_j) + z \cdot \left(g(\hat{\Lambda}_j) - c\right) \\ &= \frac{1}{\Lambda} \sum_{j=1}^{N} \hat{\Lambda}_j \cdot T_j + z \cdot \left(\sum_{j=1}^{N} \hat{\Lambda}_j - \Lambda\right). \end{aligned} \tag{6}$$

By solving the equations resulting from partial derivatives of all $\hat{\Lambda}_j$ $(1 \leq j \leq N)$ and z, the input arrival rate of each cluster is calculated based on Eq. (7):

$$\hat{\Lambda}_j = \frac{(1-\rho_j)}{\theta_j} - \frac{1}{\theta_j}\sqrt{\frac{(1-\rho_j)(\omega_j(1-\rho_j)) + \theta_j\lambda_j\mu_j}{2\theta_j(1-\rho_j)z + (\omega_j - 2\theta_j^2)}}. \tag{7}$$

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

$$\sum_{j=1}^{N} \frac{1}{\theta_j}\sqrt{\frac{(1-\rho_j)(\omega_j(1-\rho_j)) + \theta_j\lambda_j\mu_j}{2\theta_j(1-\rho_j)z + (\omega_j - 2\theta_j^2)}} = \left(\sum_{j=1}^{N} \frac{(1-\rho_j)}{\theta_j}\right) - \Lambda. \tag{8}$$

In fact, Eq. (8) expresses the relation between different parameters of the system in which j is unknown. By solving Eq. (8) for all clusters and working out z, Eq. (7) can be solved. However, finding a generic closed form solution for z in Eq. (8) is impossible [24]. Nonetheless, z can be found in the range of $[lb, ub]$ numerically. For this purpose, considering that $\hat{\Lambda}_j \geq 0$ and from Eq. (7) we can infer that:

$$z \geq \frac{\lambda_j\mu_j}{2(1-\rho_j)^2} + \frac{\theta_j}{(1-\rho_j)}. \tag{9}$$

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

$$lb = \max_{j=1}^{N}\left(\frac{\lambda_j\mu_j}{2(1-\rho_j)^2} + \frac{\theta_j}{(1-\rho_j)}\right). \tag{10}$$

If we define $\phi_j(z)$ according to Eq. (11):

$$\phi_j(z) = \frac{1}{\theta_j}\sqrt{\frac{(1-\rho_j)(\omega_j(1-\rho_j)) + \theta_j\lambda_j\mu_j}{2\theta_j(1-\rho_j)z + (\omega_j - 2\theta_j^2)}} \tag{11}$$

and considering Eq. (8), then we have:

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

The upper bound can also be worked out based on Eq. (13). ub can be reached by doubling lb up until the following condition is met

$$\sum_{j=1}^{N} \phi_j(ub) \leq \left(\sum_{j=1}^{N} \frac{(1-\rho_j)}{\theta_j}\right) - \Lambda. \tag{13}$$

If the condition in Eq. (12) is not met, then we have to decrease lb by removing clusters which are heavily loaded. The load of cluster j is comprised of local requests that have been received and external requests which are already assigned to the cluster. The load can be calculated as follows:

$$\psi_j = \frac{\lambda_j\mu_j}{2(1-\rho_j)^2} + \frac{\theta_j}{(1-\rho_j)}. \tag{14}$$

For the sake of simplicity, in Eq. (15) we have assumed that $\psi_1 \leq \psi_2 \cdots \leq \psi_N$

$$\sum_{j=1}^{k} \phi_j(\psi_k) \geq \left(\sum_{j=1}^{k} \frac{(1-\rho_j)}{\theta_j}\right) - \Lambda. \tag{15}$$

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

4. QoS and preemption-aware scheduling

In this section, we propose a workload allocation policy and a dispatch policy. The positioning of this scheduling policy in IGG is demonstrated in Fig. 2. The proposed scheduling policy comprises of two parts. The first part, discusses how the analysis mentioned in the previous section can be adapted as the workload allocation policy for external requests in IGG. The second part, is a dispatch policy which determines the sequence of dispatching external requests to different clusters considering the type of external requests.

4.1. Workload allocation policy

The analysis provided in Section 3 was based on some widely used assumptions. Here, we state these assumptions and discuss if they are valid in the Grid scenario. In the analysis provided in Section 3 we assumed that:

- Each cluster was an $M/G/1$ queue.
- All requests needed one VM (i.e. they were sequential).
- Each queue was run in FCFS fashion.
- External requests were type-less (no superiority between external requests).

On the other hand, in our scenario we encounter parallel requests (requests require more than one VM) that follow a general distribution. Additionally, we apply a conservative backfilling [41] policy as the local scheduler of each cluster. The reason of using conservative backfilling is that it increases the number of requests getting served at each moment with respect to the FCFS policy [36]. Moreover, it is proved that conservative backfilling performs better in multi-cluster environments compared with other scheduling policies [32]. Given M_j processing elements in cluster j, and $\bar{v}_j$ the average number of VMs required by external requests, the number of simultaneous requests that are served within cluster j is approximately $I_j \simeq M_j/\bar{v}_j$. We can infer that the queuing model of cluster j is $G/G/I_j$.

However, in the analyses of Section 3 we applied the $M/G/1$ queuing model instead of $G/G/I_j$. This approximation can be justified by the fact that if we consider the normalized response time in Eq. (3), then the proportion of external workload to each cluster remains unchanged. In other words, scaling up or down of the service times in the clusters, does not change the proportion of external requests allocated to each cluster. In Section 5, we validate this approximation through extensive simulations in the context of workload allocation policy for a virtualized Grid system.

Considering the above differences, we do not expect that the preemption-aware workload allocation policy still performs optimally. In fact, we examined how efficient the proposed analysis would be in a virtualized Grid environment by relaxing these assumptions.

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

$$\theta_j = \frac{\bar{v}_j \cdot \bar{d}_j}{\sum_{i=1}^{M_j} s_{ij}} \tag{16}$$

$$\tau_j = \frac{\bar{\zeta}_j \cdot \bar{\varepsilon}_j}{\sum_{i=1}^{M_j} s_{ij}} \tag{17}$$

where $\bar{\zeta}_j$ and $\bar{\varepsilon}_j$ show the average number of VMs needed and average duration of local requests. Also, $\sum_{i=1}^{M_j} s_{ij}$ indicates the

overall computing power offered by different processing elements within the cluster j. Nonetheless, if the processing elements of a cluster are homogeneous $\sum_{i=1}^{M_j} s_{ij}$ turns to $M_j \cdot s_{ij}$.

The second moment of the service time for both local and external requests are also accordingly changed. We use the coefficient of variance (CV = StDev/Mean) to obtain the modified second moment. Assuming that CV is given, the second moment of service time for external and local requests on cluster j is calculated according to Eqs. (18) and (19), respectively.

$$\omega_j = (\alpha_j \cdot \theta_j)^2 + \theta_j^2 \tag{18}$$

$$\mu_j = (\beta_j \cdot \tau_j)^2 + \tau_j^2 \tag{19}$$

where α_j and β_j show the CV of external requests and CV of local requests service time on cluster j.

The preemption-aware workload allocation policy (PAP) is presented in the form of pseudo-code in Algorithm 1. According to Algorithm 1, at first ψ is calculated for all clusters. Then, in steps 4–10, 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 until the condition defined in Eq. (15) (step 7) is met. ub is found by starting from $2 \cdot lb$ and is doubled until the condition in step 13 is met. Steps 16–21 show the bisection algorithm mentioned in Section 3 to find a 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.

It is worth mentioning that, in practice, IGG can get the required parameters for this policy by analyzing clusters' workloads. Such parameters have been used in similar research works undertaken [17,9,43].

4.2. Dispatch policy

The algorithm proposed in the previous subsection determines the routing probability to each cluster (i.e. $\hat{\Lambda}_j/\Lambda$). However, it does not offer any deterministic sequence for dispatching each external request to the clusters (i.e. dispatching Grid requests is memoryless). 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 plan to 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 the workload allocation policy. The only difference is in the sequence of requests dispatched to each cluster. For this purpose, we adapt a Billiard strategy [18] as the dispatching policy.

The Billiard strategy is the generalized form of Round Robin and considers the sequence of routing, which is called the Billiard sequence. Suppose that a billiard ball bounces in an n-dimensional cube where each side and opposite side are assigned by an integer value in a range of $\{1, 2, \ldots, n\}$. Then, the billiard sequence is generated by a series of integer values which show the sides hit by the ball when shot. This sequence is deterministic, and is different from the sequence of probabilistic scheme, which is entirely random.

Hordikj [18], proposed a method to implement this scheme and generate the billiard sequence as follows:

$$j_s = \min_{\forall j} \left\{ \frac{X_j + Y_j}{P_j} \right\} \tag{20}$$

Algorithm 1: Preemption-aware workload allocation Policy (PAP).

Input: $\bar{\Lambda}_j, \theta_j, \omega_j, \lambda_j, \tau_j, \mu_j$, for all $1 \le j \le N$.
Output: $(\hat{\Lambda}_j)$ load distribution of external requests to different clusters, for all $1 \le j \le N$.

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** $\sum_{j=1}^{k} \phi_j(\psi_k) \ge \left(\sum_{j=1}^{k} \frac{(1-\rho_j)}{\theta_j} \right) - \Lambda$ **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** $\sum_{j=1}^{k} \phi_j(z) \ge \left(\sum_{j=1}^{k} \frac{(1-\rho_j)}{\theta_j} \right) - \Lambda$ **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 $\quad$ $\hat{\Lambda}_j = 0$;

where j_s is the *target* queue, and Y and X are vectors of integers with size n. Y_j keeps track the number of requests that have been sent to the queue j. X_j reflects which queue is fastest, and is set to one for the fastest queue and zero for all other queues [2]. Y_j has to be initialized to zero, and after finding the target queue, it must be updated as $Y_{j_s} = Y_{j_s} + 1$. P_j is the fraction of external requests that are sent to the queue j and is worked out as the result of workload allocation policy in Section 4.1.

It is worth mentioning that 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 [33]. Given this policy for the local schedulers of each cluster, more valuable leases would be preempted if and only if there are no (sufficient) leases of less valuable request types to be preempted. We can infer that the 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 2.

In Algorithm 2, at first the fastest cluster is found based on the average service time for external requests in each cluster (step 1).

Algorithm 2: Request Type Dispatch Policy (RTDP).

Input: P_j, θ_j for all $1 \leq j \leq N$.
Output: *Selected Cluster* (j_s)
1 *fastestCluster* ← `findFastestCluster`(θ);
2 **foreach** *Cluster j* **do**
3 $X_j \leftarrow 0$;
4 **foreach** *RequestType i* **do**
5 $P_j^i \leftarrow P_j * \texttt{GetProportion}(i)$;
6 $Y_j^i \leftarrow 0$;
7 $X_{fastestCluster} \leftarrow 1$;
8 **foreach** *external request received* **do**
9 $i \leftarrow \texttt{GetRequestType}()$;
10 min ← *MaxValue*;
11 **foreach** *Cluster j* **do**
12 **if** $(P_j^i \neq 0)$ **then**
13 $D = (X_j + Y_j^i)/P_j^i$;
14 **if** $(D <$ min$)$ **then**
15 min ← D;
16 *tmpCluster* ← j;
17 $Y^i_{tmpCluster} \leftarrow Y^i_{tmpCluster} + 1$;
18 $j_s \leftarrow tmpCluster$;

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

5. Performance evaluation

In this section, we discuss different performance metrics considered, the scenario in which the experiments are carried out; finally, experimental results obtained from the simulations are discussed.

5.1. Performance metrics

5.1.1. User satisfaction

As mentioned earlier, both resource owners and users benefit from fewer VM preemptions. From the resource owner perspective, less VM preemption leads to less overhead on the underlying system and improves the utilization of resources. However, from the external user perspective, preemption 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 the lease. Since users of different lease types have distinct expectations from the system, it is not easy to propose a common criterion to measure the satisfaction of different users. Nonetheless, in all types of leases external users suffer from VM preemption. Therefore, we believe that the number of VM preemptions in the system is a generic metric to express both external users' and resource owners' satisfaction.

As one of the contributions of this paper is giving more precedence to more valuable external users, we also investigate how distinct scheduling policies affect more valuable leases (i.e. DC leases). To this end, for Migratable leases we consider migration rate (percentage of Migratable leases that get migrated) and for Non-Preemptable leases we consider rejection rate (percentage of Non-preemptive leases that are rejected).

5.1.2. Resource utilization

Time overhead due to VM preemptions leads to resource under-utilization. Thus, from the system owner perspective, we are interested to see how different scheduling policies affect the resource utilization. Resource utilization for the Grid system is defined as follows:

$$\text{Utilization} = \left(1 - \frac{\sum_{j=1}^{N} \text{overhead}_j}{\sum_{j=1}^{N} \text{computationTime}_j}\right) \cdot 100 \tag{21}$$

where:

$$\text{computationTime}_j = \sum_{i=1}^{|L|} v(l_i) \cdot d(l_i) \tag{22}$$

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

5.1.3. Average weighted response time (AWRT)

Preemption-based scheduling policies are usually prone to long response time for BE requests (i.e. Suspendable and Cancelable). Therefore, in our study we are interested in AWRT metric to see how the investigated scheduling policies affect response time of BE requests. Smaller values of AWRT indicate more (external) user satisfaction.

In fact, this metric measures the amount of time on average a BE lease should wait beyond its ready time to be completed. AWRT in each cluster is calculated based on the Eq. (23) [15].

$$\text{AWRT}_j = \frac{\sum_{l \in \Delta_j} v(l) \cdot d(l) \cdot (c_l - b_l)}{\sum_{l \in \Delta_j} v(l) \cdot d(l)} \tag{23}$$

where, Δ_j is the set of BE leases on cluster j. c_l and b_l show completion time and ready time, $v(l)$ and $d(l)$ represent number of VMs and duration for lease l, respectively. Then, AWRT over all clusters is defined as follows:

$$\text{AWRT} = \frac{\sum_{j=1}^{N} (M_j \cdot \text{AWRT}_j)}{\sum_{j=1}^{N} M_j}. \tag{24}$$

5.2. Experimental setup

We use GridSim [40], a discrete event simulator, to evaluate the performance of the scheduling policies. We consider a Grid with 3 clusters with 64, 128, and 256 processing elements with different computing speeds ($s_1 = 2000, s_2 = 3000, s_3 = 2100$ MIPS). This means that in the experiments we assume computing speed homogeneity within each cluster. This assumption helps us to concentrate more on the preemption aspect of resource provisioning. Moreover, considering that the resources are provisioned in the form of VMs, the assumption of homogeneous resources within the clusters is not far from reality [29]. It is worth noting that the analysis provided in Sections 3 and 4 are generic and do not consider homogeneity within cluster nodes. The cluster sizes are selected in accordance with the average demand of the current scientific applications [19]. Each cluster is managed by an LRM and a conservative backfilling scheduler. Clusters are interconnected using a 100 Mbps network bandwidth. We assume all processing elements of each cluster as a single core CPU with one VM. The maximum number of VMs in the generated requests of each cluster does not exceed the number of processing elements in that cluster. We consider size of each VM as 1024 MB [42].

The overhead time imposed by preempting VMs varies based on the type of external leases involved in preemption [39]. 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 required for suspending or migrating leases and can be ignored [39]. In our experiments, suspension time (t_s) and resumption time (t_r) are considered as 160 and 126 s, respectively [39]. The time overhead for migrating a VM with similar configuration is 372.5 s [39,42].

5.2.1. Baseline policies

To the best of our knowledge, there is no elaborate policy in the literature to be compared with proposed policies. Therefore, we evaluate the proposed policy against other basic policies which have been used as a benchmark in similar works [16]. These policies are described below:

- Round Robin (RR): In this policy IGG distributes external requests between different clusters in a round-robin fashion with a deterministic sequence. Formally, this policy is demonstrated as follows:

$$\hat{\Lambda}_j = \frac{\Lambda}{N}. \tag{25}$$

- Least Rate First (LRF): In this policy the routing probability to each cluster has an inverse relation with arrival rate of local requests to that cluster. Hence, IGG distributes the external requests with a random sequence between clusters. In other words, clusters that have a larger rate of incoming local requests would be assigned a smaller number of external requests by IGG. A formal presentation of the policy is as follows:

$$\hat{\Lambda}_j = \left(1 - \frac{\lambda_j}{\sum_{j=1}^{N} \lambda_j}\right) \cdot \Lambda. \tag{26}$$

- Biggest Cluster First (BCF): This policy is also frequently used in distributed systems [16]. In this policy the external requests are submitted to a cluster with a probability proportional to its processing capability. This policy can be formally described as follows:

$$\hat{\Lambda}_j = \left(\frac{\sum_{i=1}^{M_j} s_{ij}}{\sum_{j=1}^{N}\sum_{i=1}^{M_j} s_{ij}}\right) \cdot \Lambda. \tag{27}$$

We have also implemented the workload allocation policy (PAP) with the following details:

- We assumed that in step 16 of Algorithm 1 the precision is 0.001 ($\epsilon = 0.001$). In fact, from the experiments we noticed that values more than 0.001 do not change the results significantly.
- In Eqs. (18) and (19), to work out the second moment of service time for local and external requests, we assumed that in all clusters $\alpha_j = 0.5$ and $\beta_j = 0.1$ (i.e. *CV* of service time for external requests is more than local requests which implies that we expect more diversity in service time of external requests).
- To have a mixture of different external request types, in each workload there is 25% of each external request type which are distributed uniformly over the generated requests.

We have implemented two dispatch policies for PAP. The first one in entirely random (PAP-RND in the experiments) and the other one which is described in Algorithm 2 (PAP-RTDP in the experiments).

5.2.2. Workload model

In the experiments conducted, the DAS-2 workload model [25] has been configured to generate a two-day-long workload of parallel requests. This workload model is based on the DAS-2 multi-cluster Grid in the Netherlands.

We intend to study the behavior of different policies when they face workloads with different characteristics. For this purpose, we change the specifications of external and local requests. Particularly, we study situations where:

- External requests have a different number of VMs: In this case for external requests, we keep average *duration* = 420 s (similar to DAS-2 [25]), average *arrival rate* = 0.15; and average *local request arrival rate* = 0.12. In fact, the local request arrival rate should not be too low (in this case few preemptions take place) and should not be too high (in this case there is no room for external requests). However, external request arrival rate should be more than local arrival rate.
- The duration of external requests vary: In this case for external requests, we keep average *number of VMs* = 5 (similar to DAS-2 [25]) and average *arrival rate* = 0.15; and average *local request arrival rate* = 0.12.
- External requests' arrival rate vary: In this case for external requests, we keep average *number of VMs* = 5 and average *duration* = 420 s; and average *local request arrival rate* = 0.12.
- Local requests' arrival rate vary: In this case for external requests, we keep average *number of VMs* = 5 and average *duration* = 420 s, and average *request arrival rate* = 0.15.

More details about the generated workloads are mentioned in Table 2. To generate these workloads, we modify parameters of DAS-2 model. As shown in Table 2, the distribution of local requests in each cluster and also the distribution of external requests arriving to IGG are independent of each other. Based on the workload characterization [25], the inter-arrival rate, request size, and request duration follow Weibull, two-stage Log-uniform, and Log-normal distributions, respectively. These distributions with their parameters are listed in Table 2.

P_{one} and P_{pow2} are probabilities of request with one VM and power of two VMs in the workload, respectively. So, the mean number of VMs required by requests is given as follows:

$$\bar{v}_j = P_{\text{one}} + 2^{\lceil r \rceil}(P_{\text{pow2}}) + 2^r\left(1 - (P_{\text{one}} + P_{\text{pow2}})\right) \tag{28}$$

where r is the mean value of the two-stage uniform distribution with parameters (l, m, h, q) as listed in Table 2 and can be found as follows:

$$r = \frac{ql + m + (1-q)h}{2}. \tag{29}$$

Additionally, the mean request duration is the mean value of the Log-normal distribution with parameters (a, b) which is given by:

$$\bar{d}_j = e^{a + \frac{b^2}{2}}. \tag{30}$$

Therefore, we are able to calculate the mean request size in Eqs. (16) and (17).

Each experiment is performed on each of these workloads separately. For the sake of accuracy, each experiment is carried out 100 times by using different workloads and the average of the results is reported. In all the reported results *CV* is less than 0.01. The results of the experiments are investigated from practical and statistical perspectives. In statistical analyses we applied Two-way ANOVA and T-student tests. In doing these tests, we have ensured the normal distribution of the underlying data and equity of variance.

Table 2
Input parameters for the workload model (C:cluster).

Input parameter	Distribution	Values	Site
No. of VMs	Log-uniform	$(l = 0.8, 2.5 \leq m \leq 3.5, h = 6, q = 0.9)$	Grid
		$(l = 0.8, m = 2.5, h = 6, q = 0.9)$	C64
		$(l = 0.8, m = 3.5, h = 7, q = 0.9)$	C128
		$(l = 0.8, m = 4.5, h = 8, q = 0.9)$	C256
Request duration	Log-normal	$(3.0 \leq a \leq 5.4, b = 1.7)$	Grid
		$(a = 5.0, b = 1.7)$	C64
		$(a = 5.35, b = 1.7)$	C128
		$(a = 5.5, b = 1.7)$	C256
Inter-arrival rate external requests	Weibull	$(3.8 \leq \alpha \leq 7.0, \beta = 0.5)$	Grid
		$(\alpha = 2.0, \beta = 0.35)$	C64
		$(\alpha = 1.6, \beta = 0.35)$	C128
		$(\alpha = 1.2, \beta = 0.35)$	C256
Average inter-arrival rate local requests	Weibull	$(\alpha = 7.0, \beta = 1.1)$	Grid
		$(0.1 \leq \alpha \leq 7.0, \beta = 0.35)$	C64
		$(0.08 \leq \alpha \leq 6.0, \beta = 0.35)$	C128
		$(0.06 \leq \alpha \leq 4.5, \beta = 0.35)$	C256
P_{one}	N/A	0.2	Grid
		0.3	All clusters
P_{pow2}	N/A	0.5	Grid
		0.6	All clusters

5.3. Experimental results

5.3.1. Number of VM preemptions

The primary goal of this paper is to express the impact of scheduling policies on the number of VMs preempted in a Grid environment. Therefore, 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 Fig. 5, the number of VMs preempted increases by increasing the average number of VMs (Fig. 5(a)), duration (Fig. 5(b)), arrival rate of external requests (Fig. 5(c)), and arrival rate of local requests (Fig. 5(d)). In all of them PAP-RTDP statistically and practically significantly outperforms other policies (two-way ANOVA results in P-value < 0.001 in all the cases).

The result of a T-test analysis between PAP-RTDP and PAP-RND in Fig. 5(a) represents a significant difference. 95% confidence interval (CI) of the average difference between these policies is (2737.97, 3896.95) where P-value < 0.001. Moreover, the 95% CI of the average difference between PAP-RND and LRF-RND is (168.2, 1561.1) (P-value $= 0.02$). This indicates that PAP-RND significantly outperforms other policies.

In Fig. 5(b), we witness a sharp decrease in PAP-RTDP when the average run time is more than 300 s. 95% CI of the average difference between PAP-RTDP and PAP-RND is (342, 2354.6) using T-test (P-value $= 0.012$). Normally, as average duration increases, less free space is available, therefore, the incoming local requests result in more preemptions. A similar issue takes place in Fig. 5(c) and 95% CI of the average difference between PAP-RTDP and PAP-RND using T-test is (380.5, 4570) and P-value $= 0.02$. This means around 60% improvement rather than LRF-RND in points more than 300. 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.

Fig. 5(c) and (d) 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. 95% CI of the average difference between these policies in Fig. 5(c) is (380.5, 4570) (P-value $= 0.02$) and in Fig. 5(d) for rates more than 0.12 is (469.12, 3826.45) (P-value $= 0.02$). As we noted, the difference between PAP-RTDP and PAP-RND is more remarkable in Fig. 5(c) than Fig. 5(d). However, in general, there is a larger difference between PAP (both RND and RTDP) and other policies in Fig. 5(d) (95% CI of the average difference between PAP-RND and LRF-RND is (230.7, 4823.9) where P-value $= 0.03$). We can conclude that workload allocation policy (PAP) has more impact where inter-arrival rate of local requests is high whereas the dispatch policy has more influence where the external requests' arrival rate is high.

Generally, the difference between PAP (specially PAP-RTDP) 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.

5.3.2. Resource utilization

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-RTDP is drastically better than other policies as depicted in Fig. 6. In Fig. 6(a), 95% CI of the average difference of utilization between PAP-RTDP and LRF-RND is (12.5, 14.7) (P-value < 0.001) and the average difference between LRF-RND and PAP-RND is (2.2, 4.6) (P-value $= 0.001$) using the T-test.

However, the difference is more substantial when the average duration or arrival rate of external requests increase (Fig. 6(b) and (c)). In Fig. 6(b), 95% CI of the average difference between PAP-RTDP and LRF-RND for the durations more than 300 s using T-test is (7.3, 14.5) (P-value $= 0.002$). Additionally, 95% CI of the average difference between LRF-RND and PAP-RND is (0.9, 7.4) with P-value $= 0.01$. Also, in Fig. 6(c), 95% CI of the average difference between LRF-RND and PAP-RTDP is (1.5, 15.1) (P-value $= 0.02$). In Fig. 6(b) and (c), the reason that LRF-RND leads to better utilization compared to PAP-RND is that LRF-RND rejects fewer requests and consequently utilizes resources more than PAP-RND (see Fig. 8(d) and (f)). Expectedly, PAP-RTDP performs better than other policies in Fig. 6(d); 95% CI of the average difference between PAP-RTDP and PAP-RND for rates more than 0.12 is (11.3, 18.3) (P-value $= 0.001$).

In Fig. 6(b), we observe that PAP-RTDP results in significantly better utilization. The first reason is that PAP workload allocation policy is applied which decreases the number of VM preemptions and consequently the overall overhead. The second reason is that PAP-RTDP is directed to prevent preempting Migratable leases (as a valuable lease) that impose significant overhead when compared with other lease types to migrate VMs. In other words, PAP-RTDP dispatches a balanced mixture of all request types to different clusters. Therefore, at the time of preemption the local scheduler

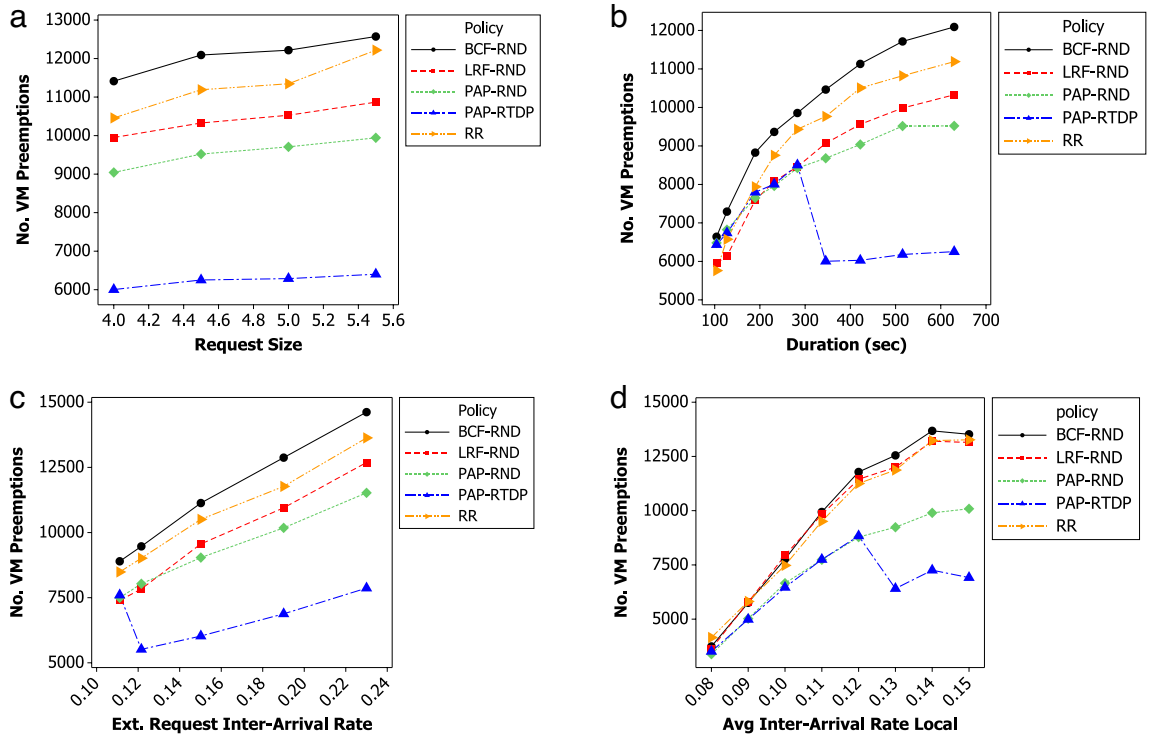

Fig. 5. The number of VMs preempted by applying different policies. The experiment is carried out by modifying (a) the average number of VMs, (b) the average duration, (c) the arrival rate of external requests, and (d) the arrival rate of local requests.

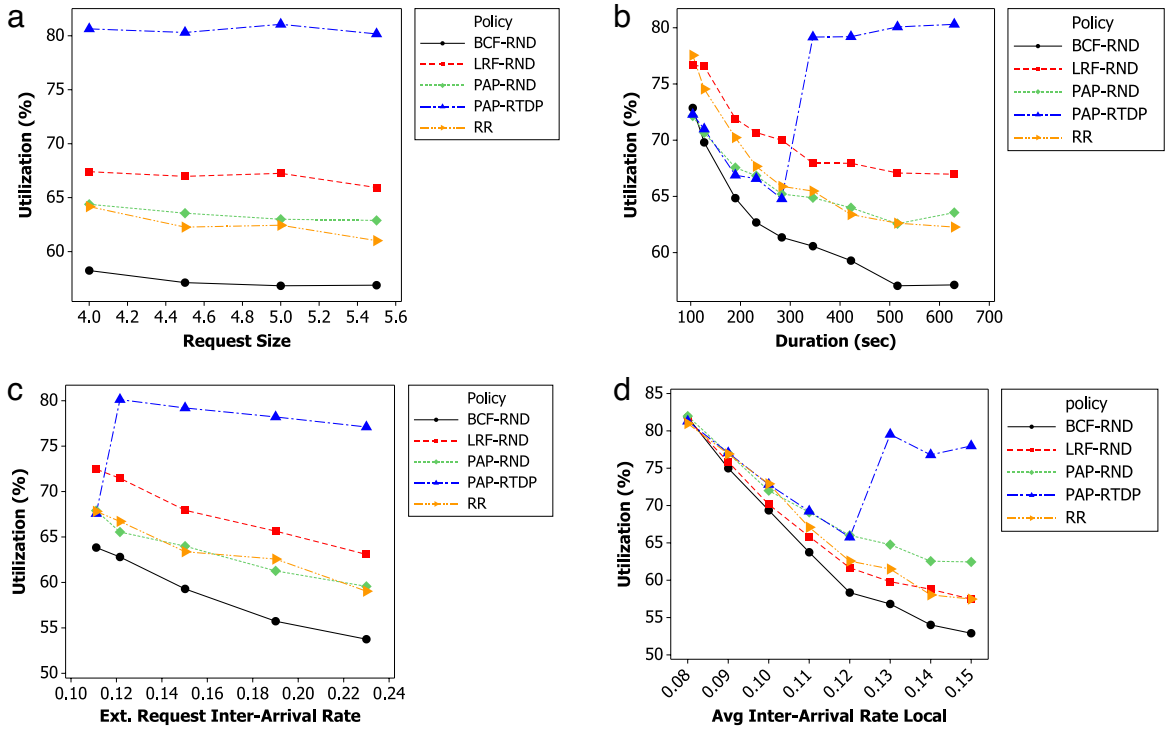

Fig. 6. Resource utilization resulted from different policies. The experiment is carried out by modifying (a) the average number of VMs, (b) the average duration, (c) the arrival rate of external requests, and (d) the arrival rate of local requests.

can preempt from BE leases that are less valuable and inherently impose less overhead to the system. In all other policies, in Fig. 6(b), resource utilization becomes more stable when external requests become longer (duration more than 300 s). The reason is that when requests become longer, 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 300 s).

5.3.3. Average weighted response time (AWRT)

In this experiment we investigate the impact of different scheduling policies on the average weighted response time for BE requests as a user centric metric.

The results of the experiment by varying different aspects of workload are shown in Fig. 7. The results prove that PAP-RND results in minimum average weighted response time compared to other policies. The reason that PAP-RTDP has longer response time than PAP-RND, is that PAP-RTDP leads to more preemptions on Cancelable and Suspendable leases. Consequently, the average response time of these requests increase.

According to the results, we can conclude that PAP-RND results in better average response time for BE requests, which implies more satisfaction of BE users. More specifically, 95% CI of the average difference between PAP-RTDP and PAP-RND in Fig. 7(a) is (3814.7, 4417) s (P-value $<$ 0.001). This difference in Fig. 7(b) is (654.1, 4158) s (P-value $=$ 0.02) for requests longer than 300 s. However, there is not a statistically significant difference for requests shorter than 300 s (P-value $=$ 0.8). 95% CI of the average difference between PAP-RTDP and PAP-RND in Fig. 7(c) is (673.8, 4753.5) s (P-value $=$ 0.02). Finally, 95% CI of the average difference between PAP-RTDP and PAP-RND in Fig. 7(d) is (1516.4, 3784) s with P-value $=$ 0.005 for rates more than 0.12.

However, in all cases, PAP-RTDP still performs significantly and practically better than other policies. More specifically, 95% CI of the average difference between PAP-RTDP and BCF-RND, in Fig. 7(b), is (1863, 6456.2) s (P-value $=$ 0.002), in Fig. 7(c) is (1665, 6115) s (P-value $=$ 0.004), and in Fig. 7(d) is (3595.8, 7661) s with P-value $<$ 0.001. The reason that BCF-RND results in lower AWRT than other policies (i.e. LRF-RND and RR), in spite of more number of preempted VMs (Fig. 5), is that according to Fig. 8, BCF-RND results in more migrations and rejections compared to other policies.

Another interesting point in this experiment is that AWRT does not change significantly by increasing the average number of VMs in the external requests (Fig. 7(a)) or their inter-arrival rate (Fig. 7(c)). The reason is that in both cases, by increasing average number of VMs of the external requests or their inter-arrival rate, more DC external requests and even more local requests get rejected. This makes more room for other requests to fit in. Therefore, although average number of VMs increase, AWRT does not increase.

5.3.4. Respecting valuable users

In this experiment we measure how different scheduling policies respect valuable users. We consider DC external requests (Migratable and Non-preemptive) as valuable users. For Migratable requests we measure the number of times that VM migration happens (migration rate). For Non-preemptive external requests, however, we consider the rejection rate as the measurement criterion. The results of the experiments are illustrated in Fig. 8.

This experiment expresses the efficacy of PAP-RTDP policy in migrating and rejecting less number of external requests. In all sub-figures of Fig. 8, we can notice that PAP-RTDP dispatching policy has substantially reduced the percentage of migrations and also rejections. Details of the 95% CI of the average differences between PAP-RTDP and PAP-RND, are presented in Table 3. According to Table 3, in almost all experiments PAP-RTDP leads to statistically and practically significant difference with PAP-RND. Except in Fig. 8(h); Fig. 8(c) and (d) where request duration is less than 300 s. P-values in these points are 0.8 and 0.7 respectively, which proves the null hypothesis (i.e. PAP-RTDP and PAP-RND are not statistically different).

Table 3
95% confidence interval (CI) of the average differences between PAP-RTDP and PAP-RND related to Fig. 8.

Figure	95% CI	P-value
8(a)	(2.4, 3.8)	$<$0.001
8(b)	(9.3, 12.7)	$<$0.001
8(c)	(2.6, 4.3)	0.001
8(d)	(5.9, 12.5)	0.003
8(e)	(0.14, 7.8)	0.04
8(f)	(2.1, 16)	0.02
8(g)	(0.02, 3.2)	0.04
8(h)	Not statistically significant	0.2

Particularly, in Fig. 8(h), although PAP-RTDP is not statistically better than PAP-RND, we observe a marginal improvement in the rejection rate mainly for rates more than 0.12. We also witness a sharp decrease both in Fig. 8(c) and (d) for requests that last more than 300 s. Basically, this is because the overall number of preemptions in that point has decreased (see Fig. 5(b)).

In Fig. 8(e) and (f) 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. The issue is observed in Fig. 8(a), (b) and that is why we notice a slight decrease mainly in Fig. 8(b).

6. Related work

There are several research works that have investigated "preemption" of jobs/requests in parallel distributed computing. Scheduling a mixture of different job/request types has also been extensively studied. Particularly, the mixture of local and external requests have been investigated [24,14,4,3]. Meta-scheduling has also been under through investigation in multi-cluster/Grid computing environments. In this section, we provide a review on the recent studies in these areas and position our work in comparison with them.

Bucur and Epema [5] have evaluated the average response time of jobs in DAS-2 [25] by applying local, global (external), and combination of these schedulers in the system. They have concluded that the combination of local and global schedulers is the best choice in DAS-2. They have also observed that it is better to give more priority to local jobs while the global jobs should also have chance to run.

Assuncao and Buyya [10] have proposed policies in IGG based on adaptive partitioning of the availability times between local and external requests in each cluster. Consequently, there is a communication overhead between IGG and clusters for submitting availability information. Additionally, for security reasons some clusters may not be willing to share their availability information. Finally, there is a possibility that the availability information will be imprecise which deteriorates the scheduling results. By contrast, our scheduling method is non-observable and does not rely on availability information of the clusters.

He et al. [17] have aimed at minimizing response time and miss-rate for non-real-time and soft real-time jobs separately in the global scheduler of a multi-cluster by applying a non-observable approach. They have also recognized workload allocation policy from job dispatching policy. Nonetheless, for job dispatching they have applied policies such as weighted round robin policies. By contrast, we consider a mixture of such jobs in our scenario which are affected by local requests in each cluster. In another work, He et al. [16] have proposed a multi-cluster scheduling policy where the global scheduler batches incoming jobs and assigns each batch to a cluster. In the next step, a local scheduler performs further

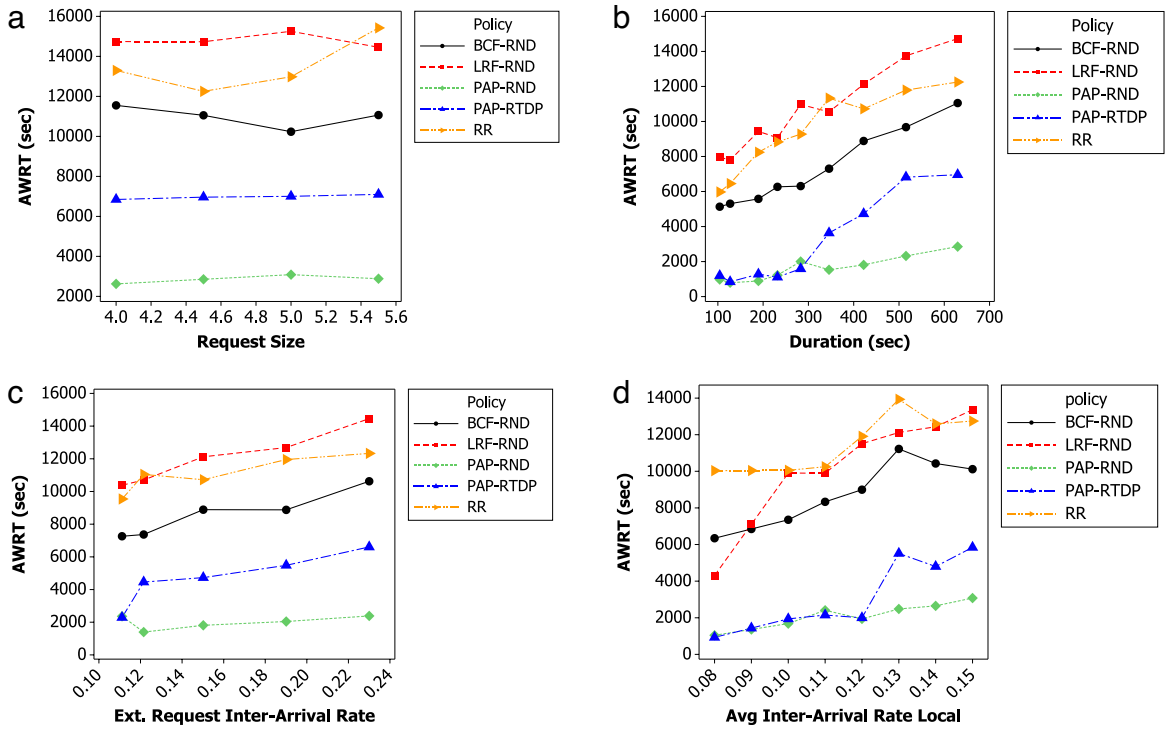

Fig. 7. Average weighted response time resulted from different policies. The experiment is carried out by modifying (a) the average number of VMs, (b) the average duration, (c) the arrival rate of external requests, and (d) the arrival rate of local requests.

tuning to run the assigned jobs with minimized make span and idle times.

Haizea [38] 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. [39], have also investigated the overhead time imposed by preempting a lease (suspending and resuming a VM) in Haizea. The similarity of our work and research carried out by Sotomayor et al. is considering a combination of best-effort and deadline-constraint requests in the system. However, we propose a scheduling policy to decrease the number of VM preemptions in the system whereas they focus on the overhead aspects of lease preemption.

Scojo-PECT [37] 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. Nonetheless, they have not considered the overhead of suspending VMs on disk in their evaluations. The main difference with our work, however, is the goal of scheduling. We aim to minimize the number of VM preemptions whereas Sodan et al.'s goal is fair share scheduling.

In [31] authors have proposed a prediction method for unavailable periods in fine-grained cycle sharing systems where there are mixture of local jobs and global (guest) jobs. In [35] a variation of multi-level feed-back scheduling policy is applied for a mixture of short and long jobs in a multi-Grid environment with the goal of minimizing response time. In [7] a multi-level reconfiguration manager is proposed in a time-shared multi-Grid where external (Grid) jobs and local jobs coexist. The reconfiguration is performed based on the local load level.

Kettimuthu et al. [21] 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 in which if the suspension factor of an idle job is sufficiently more than the running jobs, then it can preempt the running job. However, the authors have not specified how to minimize the number of preemptions; instead, they decide when to do the preemption. The proposed policy is starvation free since it operates based on the response ratio of jobs.

Amar et al. [1] have added preemption to cope with the non-optimality of the on-line scheduling policies. In the preemption policy jobs are prioritized based on their remaining time as well as the job's waiting cost. Schwiegelshohn and Yahyapour [34] have investigated different variations of preemptive first-come-first-serve policy for an on-line scheduler that schedules parallel jobs where the jobs' run times and end times are unknown. Margo et al. [28] have leveraged a priority scheduling based on preempting jobs for Catalina (San Diego Super-Computer scheduler) to increase the utilization of the resources. They determine the priority of each job based on the expansion factor and number of processing elements each job needs.

7. Conclusions and future work

In this research we explored how we can minimize the side-effects of VM preemptions in a federation of virtualized Grids such as InterGrid. We consider circumstances that local requests in each cluster of a Grid coexist with external requests. Particularly, we consider situations that external requests have different levels of QoS (i.e. some external requests are more important than others). For this purpose, we proposed a preemption-aware workload allocation policy (PAP) in IGG to distribute external requests amongst different clusters in a way that minimizes the overall number of VM preemptions that take place in a Grid. Additionally, we investigated on a dispatch policy that regulates dispatching of different external requests in a way that external requests with higher QoS requirements (more valuable users) have less chance of getting preempted. The proposed policies are knowledge-free and do not impose any communication overhead to the underlying system.

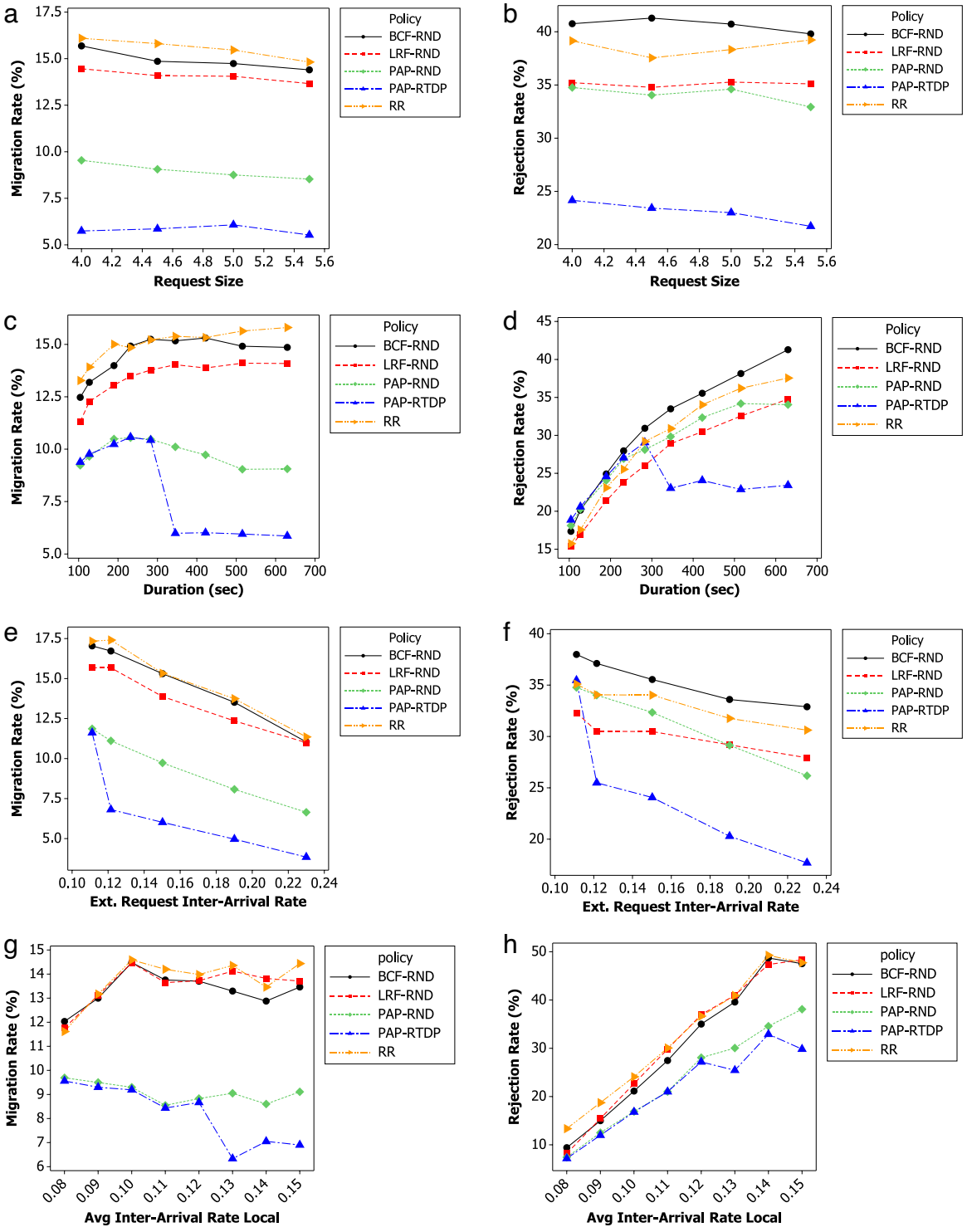

Fig. 8. Respecting deadline constraint (more valuable) users resulted from different policies. The experiment is carried out by modifying (a), (b) the average number of VMs, (c), (d) the average duration, (e), (f) the arrival rate of external requests, and (g), (h) the arrival rate of local requests.

We compared the performance of the proposed policies with variety of other policies. The results of the experiments indicate the PAP-RND and specifically PAP-RTDP significantly decreases the number of VM preemptions which in turn affects both user centric and system centric criteria. We observed that PAP-RTDP resulted in at least 60% improvement in VM preemptions compared with other policies. This decrease in number of VM preemptions improves the utilization of the resources and decreases average weighted response time of the external requests (by more than 50%). PAP-RTDP, particularly, is better for more valuable external requests and effectively results in less VM preemption for valuable external requests. Although PAP-RTDP in general preempts fewer number of VMs, PAP-RND results in better average response time for BE requests. In fact, in the case of PAP-RTDP, since most of the preempted leases are BE, it does not result in minimum average response time. This indirectly represents the efficacy of PAP-RTDP

for more valuable external requests. We also noticed that changing request size is not that effective on the performance of scheduling policies.

Although we carried out this research in the context of federated Grids, we believe that it is extensively applicable in other lease-based Grid/Cloud resource providers where requests with higher priority (such as local or more valuable requests) coexist with other requests. One conceivable application is in Cloud providers (IaaS providers) where there are certain priorities between different users; and resource owners tend to minimize the number of VM preemptions both for more user satisfaction and also system utilization purposes. For instance, Spot instances in Amazon EC2 can be canceled if the price goes beyond the user bid. However, by applying an appropriate preemption-aware scheduling policy Amazon can minimize the number of preemptions which causes more (spot) user satisfaction and, at the same time, more resource utilization and even more revenue for the company.

There are several avenues of future works that could be pursued as the extensions of this research. One interesting issue is how IGG can decrease the response time for BE requests in its scheduling. Another interesting extension would be considering co-allocation of the incoming external requests on different clusters to further decrease the number of VM preemptions.

References

[1] L. Amar, A. Mu'Alem, J. Stober, The power of preemption in economic online markets, in: Proceedings of the 5th International Workshop on Grid Economics and Business Models, GECON'08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 41–57.

[2] J. Anselmi, 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, pp. 1–8.

[3] N.T. Argon, L. Ding, K.D. Glazebrook, S. Ziya, Dynamic routing of customers with general delay costs in a multiserver queuing system, Probability in the Engineering and Informational Sciences 23 (2) (2009) 175–203.

[4] F. Bonomi, A. Kumar, Adaptive optimal load balancing in a nonhomogeneous multiserver system with a central job scheduler, IEEE Transactions on Computers 39 (1990) 1232–1250.

[5] A. Bucur, D. Epema, Priorities among multiple queues for processor co-allocation in multicluster systems, in: Proceedings of the 36th Annual Symposium on Simulation, IEEE Computer Society, 2003, pp. 15–22.

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

[7] P. Chen, J. Chang, T. Liang, C. Shieh, A progressive multi-layer resource reconfiguration framework for time-shared Grid systems, Future Generation Computer Systems 25 (6) (2009) 662–673.

[8] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, M. Bowman, PlanetLab: an overlay testbed for broad-coverage services, ACM SIGCOMM Computer Communication Review 33 (3) (2003) 3–12.

[9] M. Colajanni, P. Yu, V. Cardellini, Dynamic load balancing in geographically distributed heterogeneous web servers, in: Proceedings 18th International Conference on Distributed Computing Systems, 1998, IEEE, 2002, pp. 295–302.

[10] M.D. De Assunção, R. Buyya, Performance analysis of multiple site resource provisioning: effects of the precision of availability information, in: Proceedings of the 15th International Conference on High Performance Computing, HiPC'08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 157–168.

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

[12] A. di Costanzo, M.D. de Assuncao, R. Buyya, Harnessing cloud technologies for a virtualized distributed computing infrastructure, IEEE Internet Computing 13 (2009) 24–33. doi:10.1109/MIC.2009.108.

[13] J. Fontán, T. Vázquez, L. Gonzalez, R.S. Montero, I.M. Llorente, OpenNebula: the open source virtual machine manager for cluster computing, in: Open Source Grid and Cluster Software Conference—Book of Abstracts, San Francisco, USA, 2008.

[14] L. Gong, X.-H. Sun, E.F. Watson, Performance modeling and prediction of nondedicated network computing, IEEE Transactions on Computers 51 (2002) 1041–1055.

[15] C. Grimme, J. Lepping, A. Papaspyrou, Prospects of collaboration between compute providers by means of job interchange, in: Job Scheduling Strategies for Parallel Processing, Springer, 2008, pp. 132–151.

[16] L. He, S. Jarvis, D. Spooner, X. Chen, G. Nudd, Dynamic scheduling of parallel jobs with QoS demands in multiclusters and Grids, in: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, IEEE Computer Society, 2004, pp. 402–409.

[17] L. He, S. Jarvis, D. Spooner, H. Jiang, D. Dillenberger, G. Nudd, Allocating non-real-time and soft real-time jobs in multiclusters, IEEE Transactions on Parallel and Distributed Systems (2006) 99–112.

[18] A. Hordijk, D. van der Laan, Periodic routing to parallel queues and Billiard sequences, Mathematical Methods of Operations Research 59 (2004) 173–192.

[19] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, D. Epema, Performance analysis of cloud computing services for many-tasks scientific computing, IEEE Transactions on Parallel and Distributed Systems (99) (2011) 1–16.

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

[21] R. Kettimuthu, V. Subramani, S. Srinivasan, T. Gopalsamy, D.K. Panda, P. Sadayappan, Selective preemption strategies for parallel job scheduling, International Journal of High Performance Computing and Networking 3 (2–3) (2005) 122–152.

[22] L. Kleinrock, Queueing Systems: Computer Applications, Wiley-Interscience, 1976.

[23] Y.-K. Kwok, I. Ahmad, Dynamic critical-path scheduling: an effective technique for allocating task graphs to multiprocessors, IEEE Transactions on Parallel and Distributed Systems 7 (5) (1996) 506–521.

[24] K. Li, Optimal load distribution in nondedicated heterogeneous cluster and Grid computing environments, Journal of Systems Architecture 54 (2008) 111–123.

[25] H. Li, D. Groep, L. Wolters, Workload characteristics of a multi-cluster supercomputer, in: Job Scheduling Strategies for Parallel Processing, JSSPP, Springer, 2005, pp. 176–193.

[26] C. Liu, S. Baskiyar, A general distributed scalable Grid scheduler for independent tasks, Journal of Parallel and Distributed Computing 69 (3) (2009) 307–314.

[27] Y. Liu, B. Veeravalli, S. Viswanathan, Novel critical-path based low-energy scheduling algorithms for heterogeneous multiprocessor real-time embedded systems, in: Proceedings of the 13th International Conference on Parallel and Distributed Systems—Volume 01, IEEE Computer Society, Washington, DC, USA, 2007, pp. 1–8.

[28] M. Margo, K. Yoshimoto, P. Kovatch, Preemption and priority calculation in production job data, in: Proceedings of the 9th LCI International Conference on High-Performance Clustered Computing, Urbana, Illinois, USA, 2008.

[29] M.A. Murphy, B. Kagey, M. Fenn, S. Goasguen, Dynamic provisioning of virtual organization clusters, in: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID'09, IEEE Computer Society, 2009, pp. 364–371.

[30] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, D. Zagorodnov, The Eucalyptus open-source cloud-computing system, in: 1st Workshop of Cloud Computing and its Applications, 2008.

[31] X. Ren, S. Lee, R. Eigenmann, S. Bagchi, Prediction of resource availability in fine-grained cycle sharing systems empirical evaluation, Journal of Grid Computing 5 (2) (2007) 173–195.

[32] G. Sabin, R. Kettimuthu, A. Rajan, P. Sadayappan, Scheduling of parallel jobs in a heterogeneous multi-site environment, in: D. Feitelson, L. Rudolph, U. Schwiegelshohn (Eds.), Job Scheduling Strategies for Parallel Processing, Berlin, Heidelberg, in: Lecture Notes in Computer Science, vol. 2862, Springer, 2003, pp. 87–104.

[33] M. Amini Salehi, B. Javadi, R. Buyya, Resource provisioning based on leases preemption in InterGrid, in: Australasian Computer Science Conference, ACSC 2011, in: CRPIT, vol. 113, ACS, Perth, Australia, 2011, pp. 25–34.

[34] U. Schwiegelshohn, R. Yahyapour, Fairness in parallel job scheduling, Journal of Scheduling 3 (5) (2000) 297–320.

[35] M. Silberstein, D. Geiger, A. Schuster, M. Livny, Scheduling mixed workloads in multi-Grids: the Grid execution hierarchy, in: 2006 15th IEEE International Symposium on High Performance Distributed Computing, IEEE, 2006, pp. 291–302.

[36] Q. Snell, M.J. Clement, D.B. Jackson, Preemption based backfill, in: Job Scheduling Strategies for Parallel Processing, JSSPP, Springer, 2002, pp. 24–37.

[37] A. Sodan, Service control with the preemptive parallel job scheduler Scojo-PECT, Journal of Cluster Computing (2010) 1–18.

[38] B. Sotomayor, K. Keahey, I. Foster, 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, 2008, pp. 87–96.

[39] B. Sotomayor, R.S. Montero, I.M. Llorente, I. Foster, 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, 2009, pp. 59–68.

[40] A. Sulistio, U. Cibej, S. Venugopal, B. Robic, R. Buyya, A toolkit for modelling and simulating data Grids: an extension to GridSim, CCPE Concurrency and Computation: Practice and Experience 20 (13) (2008) 1591–1609.

[41] W. Tang, N. Desai, D. Buettner, Z. Lan, Analyzing and adjusting user runtime estimates to improve job scheduling on the Blue Gene/P, in: IEEE International Symposium on Parallel & Distributed Processing, IPDPS, 2010, IEEE, pp. 1–11.

[42] M. Zhao, R. Figueiredo, 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, 2007, pp. 5–11.

[43] S. Zhou, A trace-driven simulation study of dynamic load balancing, IEEE Transactions on Software Engineering 14 (9) (2002) 1327–1341.

Mohsen Amini Salehi is a Ph.D. student under the supervision of professor Rajkumar Buyya in CLOUDS lab, Melbourne University, Australia. He was a university lecturer in Azad University of Mashhad, Iran in 2006–2008. He received his M.Sc. from Ferdowsi University of Mashhad and B.Sc. from Azad University of Mashhad in Software Engineering in 2006 and 2003, respectively. His thesis for his M.Sc. was on load balancing in Grid computing. Currently, he is involved in the InterGrid project and he works on preemption-aware scheduling methods in virtualized resource providers.

Bahman Javadi is a Research Fellow at the University of Melbourne, Australia. He was a postdoctoral fellow in the MESCAL team at INRIA Rhone-Alpes, France in 2008–2010. He received his MS and Ph.D. in Computer Engineering from Amirkabir University of Technology in 2001 and 2007, respectively. He has been working as a research scholar in the School of Engineering and Information Technology, Deakin University, Australia from 2005–2006. He is co-founder of the Failure Trace Archive, which serves as a public repository of failure traces and algorithms for distributed systems. He served as a program committee of many international conferences and workshops and co-guest editor of a special issue of the Journal of Future Generation Computer Systems on Desktop Grids. His research interests include Cloud and Grid computing, performance evaluation of large scale distributed computing systems, and reliability and fault tolerance.

Dr. Rajkumar Buyya is Professor of Computer Science and Software Engineering; and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft Pty Ltd., a spin-off company of the University, commercializing its innovations in Grid and Cloud Computing. He has authored and published over 300 research papers and four text books. The books on emerging topics that Dr. Buyya edited include, High Performance Cluster Computing (Prentice Hall, USA, 1999), Content Delivery Networks (Springer, Germany, 2008), Market-Oriented Grid and Utility Computing (Wiley, USA, 2009), and Cloud Computing: Principles and Paradigms (Wiley, USA, 2011). He is one of the highly cited authors in computer science and software engineering worldwide (h-index = 52, g-index = 111, 14 500 citations).

Software technologies for Grid and Cloud computing developed under Dr. Buyya's leadership have gained rapid acceptance and are in use at several academic institutions and commercial enterprises in 40 countries around the world. Dr. Buyya has led the establishment and development of key community activities, including serving as foundation Chair of the IEEE Technical Committee on Scalable Computing and four IEEE conferences (CCGrid, Cluster, Grid, and e-Science). He has presented over 250 invited talks on his vision on IT Futures and advanced computing technologies at international conferences and institutions in Asia, Australia, Europe, North America, and South America. These contributions and international research leadership of Dr. Buyya are recognized through the award of "2009 IEEE Medal for Excellence in Scalable Computing" from the IEEE Computer Society, USA. Manjrasoft's Aneka technology for Cloud Computing developed under his leadership has received "2010 Asia Pacific Frost and Sullivan New Product Innovation Award".