2012 26th IEEE International Conference on Advanced Information Networking and Applications

Preemption-aware Admission Control in a Virtualized Grid Federation

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
{mohsena,bahmanj,raj}@csse.unimelb.edu.au

***Abstract*—Many applications in federated Grids have quality-of-service (QoS) constraints such as deadline. Admission control mechanisms assure QoS constraints of the applications by limiting the number of user requests accepted by a resource provider. However, in order to maximize their profit, resource owners are interested in accepting as many requests as possible. In these circumstances, the question that arises is: what is the effective number of requests that can be accepted by a resource provider in a way that the number of accepted external requests is maximized and, at the same time, QoS violations are minimized. In this paper, we answer this question in the context of a virtualized federated Grid environment, where each Grid serves requests from external users along with its local users and requests of local users have *preemptive priority* over external requests. We apply analytical queuing model to address this question. Additionally, we derive a preemption-aware admission control policy based on the proposed model. Simulation results under realistic working conditions indicate that the proposed policy improves the number of completed external requests (up to 25%). In terms of QoS violations, the 95% confidence interval of the average difference with other policies is between (14.79%,18.56%).**

I. Introduction

Large scale shared computing resources, such as federated Grids, serve different types of users with diverse performance expectations. Specifically, many applications in these environments cannot tolerate performance variations and need to secure strict quality-of-service (QoS) for their users. Applications such as media servers [15] and individualized patient treatment [14], [19], are instances of QoS-constrained applications.

A federated Grid environment, as a large scale resource sharing system, aims to provide a framework that interconnects islands of Grids [4] and facilitates executing applications with heavy computational requirements, which can potentially have QoS constraints. Recently, resources in such environments are provisioned in the form of Virtual Machines (VMs), which allow different types of users to create execution environments for their applications. GridWay [22] and specifically InterGrid [4] are instances of such federated Grid environments.

As illustrated in Figure 1, in each constituent Grid of a federated Grid environment, the provisioning rights over several resource providers (e.g. clusters) inside the Grid are delegated to a gateway (GW). The gateways coordinate resource allocation for requests coming from other Grids (external users) through contracts between them [4]. On the other hand, local users of each resource provider (cluster) send their requests directly to the local resource manager (LRM) of the cluster.

Typically, local requests have *priority* over external requests in each cluster [7]. In other words, the organization that owns the resources would like to ensure that its local users (hereafter termed *local requests*) have priority access to the resources. In this situation, external users (hereafter termed *external requests*) are welcome to use resources if they are available. Nonetheless, external requests should not delay the execution of local requests.

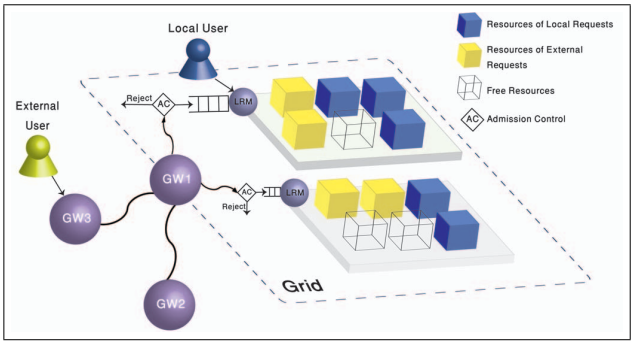

Fig. 1. Illustration of distinct users (i.e. local and external requests) in a federated Grid where each Grid is composed of several clusters.

To assure that the local requests of a cluster can get access to the resources, in our previous research [2], we leveraged preemption of external requests in favor of local requests. However, preemption leads to increase in response time of requests. Especially, when the external requests are deadline constrained, preemption increases the likelihood of deadline violation.

Deadline violations of external requests become more critical as a cluster is over-subscribed to the external requests (i.e. in Figure 1 the queuing time of external requests in the LRM gets long). In general, there are several approaches in resource sharing environments, to meet deadline constraints for applications. One common approach is applying *admission control* mechanisms that prevents the resources from becoming over-subscribed and, as a result, deadline violations are avoided.

On the other hand, resource owners are interested to accept as many external requests as possible which potentially lead

1550-445X/12 $26.00 © 2012 IEEE
DOI 10.1109/AINA.2012.52

to more deadline violations. Therefore, the question that arises is: what is the ideal number of external requests a cluster can accept in a way that the number of accepted external requests in a cluster is maximized while the QoS violation is avoided?

Previous research works on admission control either have been in non-prioritized environment, or did not consider the impact of preempting VMs [18], [1], [12], [23]. However, our work considers these issues and finds out the number of external requests (i.e. queue length) for each cluster.

More specifically, in this paper, we propose a preemption-aware admission control policy within the LRM of each cluster. This policy determines the suitable number of external requests that can be accepted within each cluster of a Grid. The objective of the policy is maximizing the number of accepted external requests as well as minimizing deadline violation for external requests. We apply analytical queuing model to address this question. The advantage of applying analytical methods is that it provides a performance model based on system parameters. Additionally, analytical methods evaluate performance metrics in a time efficient manner and, therefore, reduce the computational time.

We carry out this research in the context of InterGrid [4] which complies with the characteristics mentioned for virtualized federated Grid systems. A detailed structure of InterGrid is discussed in the next section.

The rest of this paper is organized as follows: In Section II, an overview of the InterGrid environment is provided. In Section III related research work is introduced. The proposed analytical queuing model is described in Section IV, which is followed by the preemption-aware admission control policy in Section IV-A. Performance evaluation of the proposed policy is reported in Section V. Finally, conclusion and future works are provided in Section VI.

II. Background and Context

As we carry out our study in the context of InterGrid, we provide a short description on its architecture and implementation in this section. Interested readers can refer to [4] for more details. The structure of different components in InterGrid is similar to that presented in Figure 1. Therefore, we refer to that figure while we describe InterGrid in this section.

InterGrid aims to provide a framework that allows users to create execution environments for various applications on top of the physical infrastructure participating in Grid systems. Peering arrangements established between gateways (termed IGG in the InterGrid context) enable the allocation of resources from different Grids to fulfill the requirements of the execution environments.

The Local Resource Manager (LRM)[1] is the resource manager in each cluster and provisions resources for the local and external requests. Virtual Machine (VM) technology is used in each cluster of InterGrid for resource provisioning. Requests in InterGrid are contiguous and have to be served within resources of a single cluster. Each request has the following characteristics:

- Type of the request (local/external);
- Duration of the request;
- Deadline of the request.

Since local requests in the clusters have high priority, external requests are preempted in favor of local requests. Preempted external requests are rescheduled in the next available time-slot. Local requests in the clusters are immediate and non-preemptive. This means that local requests require resources as soon as they are submitted to the LRM and when they get access to the resources they will be running until their completion.

The central element of the IGG is the scheduler, which implements provisioning policies and peering with other IGGs. The scheduler performs creation, starting, and stopping of VMs through the Virtual Machine Manager (VMM). The VMM implementation is generic, so different LRMs can interact with it. Currently, it is possible for the VMM to interact with OpenNebula [5] and Eucalyptus [16] as different LRMs. In addition, two interfaces to connect to a Grid middleware (i.e., Grid'5000) and a Cloud IaaS provider (i.e., Amazon EC2[2]) have been developed.

III. Related Work

Significant research work has been done on admission control in Grid computing. Much research has also been undertaken on preempting jobs/VMs in distributed systems. In this section, we provide a review on the recent research in these areas and position our work in relation with them.

Sandholm et al. [18] investigated how admission control can increase user fulfillment in a computational market. Specifically, they considered the mixture of best-effort (to improve resource utilization) and QoS-constrained jobs (to improve revenue). They also leveraged partial preemption of best-effort requests. The admission control policy accepts a new request if the current requests can still respect their QoS. Therefore, the feasibility test should be done for each incoming request. Conversely, our work focuses on finding out the ideal queue length and does not impose overhead on the system for each incoming request.

Almeida et al. [1] proposed an optimization model for admission control and resource allocation in a virtualized web server that considers owners revenue and cost while guarantees the response time of requests. They considered several request classes and with a certain probability guarantee that the response time of each class will not be more than what is agreed in the SLA. In their analysis they consider the Poisson distribution for arrival rate and exponential service rate. Our research differs from this work in that we consider general distribution for service time in our analysis (which is more realistic based on the available workloads [10]).

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

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

Additionally, we consider circumstances when preemption takes place whereas they do not assume preemption in their work.

Xavier et al. [23] applied an admission control policy for shared resources where some large jobs takes precedence over many small jobs that are waiting in the queue. Resource providers determine the resource prices based on the degree of contention and instantaneous utilization of resources. Consumers also bid for the resources based on their budget. In general, a job can get a resource if it can compensate the loss of earning resulting from not admitting several small jobs. On the contrary, we study a situation where preemption happens and requests also have different priorities.

QoPS is a scheduler that provides QoS (response time) guarantee for parallel jobs in a distributed system [12]. In QoPS the cost of running a job depends on the amount of resource usage as well as responsiveness of the job. For this purpose, QoPS makes reservation for jobs and reschedules other jobs in a way that their deadlines are met. In our research we deal with the number of requests that can be accepted, whereas policies such as QoPS determines the scheduling/rescheduling of the accepted requests within the queue.

A performance model is provided to work out the run time of an external task in a single processor of a Network of Workstations (NOW) [7] where local and external requests coexist and local tasks have preemptive priority over external tasks. Although the considered scenario is similar to the scenario we consider, the main difference is that we take into account several parallel external requests whereas Gong et al. have considered one sequential external request at any moment. In other words, there is no queue for external requests in [7] whereas our research focuses on the number of external requests that can be admitted without violating their deadlines.

IV. Analytical Queuing Model

In this section, we describe the analytical model to find the ideal queue length for external requests within the LRM of each cluster in the federated Grid environment. This section is followed by the proposal of an admission control policy, built upon the analytical model provided. Table I gives the list of symbols we use along with their meanings.

It is worth noting again that submitted local requests to a cluster have to be executed immediately unless the requested resources are occupied by other local requests. On the other hand, an external request that is running within a cluster is preempted to free space for the local request. In this case, the preempted external request is rescheduled on the next available time-slot.

The queuing model that represents a gateway along with several non-dedicated clusters (i.e. clusters with shared resources between local and Grid requests) is depicted in Figure 2. According to this figure, cluster j receives requests from two independent sources. One source is a stream of local requests with mean arrival rate λ_j and the other is a stream of external requests, which are sent by the gateway with mean arrival rate Λ_j.

TABLE I
Description of symbols used in the queuing model.

Symbol	Description
$E(W_j)$	Expected waiting time of external requests in the LRM queue cluster j
$E(T_j)$	Expected service time of external requests in cluster j
$E(R_j)$	Expected response time of external requests in cluster j
D	Expected deadline of external requests
Λ_j	Arrival rate of external requests to cluster j
μ_l^j	Service rate of local requests in cluster j
μ_e^j	Service rate of external requests in cluster j
ω	Mean duration of external requests
λ_j	Arrival rate of local requests to cluster j
ρ_e^j	Λ_j/μ_e^j
ρ_l^j	λ_j/μ_l^j
α	Scale parameter in Gamma distribution
β	Shape parameter in Gamma distribution
θ_j	Coefficient of Variance (CV) for service time of local requests in cluster j
e_i^j	ith running slice for an external request in cluster j
l_i^j	Mean duration of local request i in cluster j
$rate_l$	Low-urgency rate
u_l	Average deadline ratio for low-urgency requests
u_h	Average deadline ratio for high-urgency requests

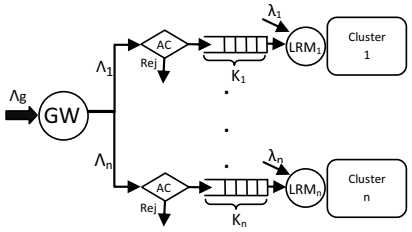

Fig. 2. Queuing model for resource provisioning in a Grid with *n* clusters.

As mentioned earlier, the analytical model aims at working out the ideal queue length for external requests in each cluster in a way that deadline violations of external requests are minimized and, at the same time, the number of completed external requests is maximized. Our analysis is based on the following assumptions:

- All requests are *moldable* applications which can spread over all available resources in a cluster. Therefore, we can assume that each cluster is a single server.
- Local requests are immediate and must be processed as soon as they are submitted. We assume an $M/G/1$ queue to model the service time of local requests.
- External requests are submitted to a queue in each cluster that can be modeled as an $M/G/1/K$ queue model.

In this situation, the analysis goal is finding out the suitable value of K_j for cluster j in a way that deadline of the external requests is not violated. Thus, our primary objective function can be expressed as follows:

$$E(R_j) = E(W_j) + E(T_j) \leq D \tag{1}$$

where $E(R_j)$ is the expected response time; $E(W_j)$ and $E(T_j)$ are the expected waiting time and expected service time for external requests in cluster j, respectively. D is the expected deadline for external requests. Over the next few paragraphs we discuss how $E(W_j)$, $E(T_j)$ can be obtained for cluster j.

If we suppose that an external request, with overall runtime ω, encounters n preemptions before getting completed,

then the service time (T) of the external request e can be formulated as follows:

$$T_j = e_1^j + l_1^j + e_2^j + l_2^j + ... + e_n^j + l_n^j + \epsilon \quad (2)$$

where l_i^j is the duration of the local request i and e_i^j is the ith running slice of the external request e in cluster j and we have $e_1^j + e_2^j + ... + e_n^j + \epsilon = \omega$. Also, ϵ is the last running slice of e. Given that the arrival rate of requests follows the Poisson distribution, we can conclude that e_i^j follows the exponential distribution and n follows Gamma distribution [13]. Therefore, we have $E(n) = \lambda_j \omega$ where λ_j is the arrival rate of local requests in cluster j. Thus, $E(T_j)$ is worked out based on Equation 3.

$$\begin{aligned} E(T_j) = E(E(T_j|n)) &= E(\omega + l_1^j + l_2^j + ... + l_n^j|n) \\ &= E(\omega + n \cdot E(l_1^j)) \\ &= \omega + \lambda_j \omega E(l_1^j) \end{aligned} \quad (3)$$

where $E(l_1^j) = 1/(\mu_l^j - \lambda_j)$ since it follows the $M/G/1$ queuing system. Hence, the expected service time and variance of service time for external requests ($E(T_j)$ and $V(T_j)$ respectively) are worked out through Equations 4 and 5 [7]:

$$E(T_j) = \frac{\mu_l^j \cdot \omega}{\mu_l^j - \lambda_j} = \frac{\omega}{1 - \rho_l^j} \quad (4)$$

$$V(T_j) = \frac{\rho_l^j}{(1-\rho_l^j)^3} \cdot \frac{\theta_j^2 + 1}{\mu_e^j} \cdot \omega \quad (5)$$

where θ_j is the coefficient of variance (CV) of service time for local requests; μ_e^j is the service rate of external requests; and ρ_l^j is the queue utilization for local requests in cluster j. According to Bose [3], the average waiting time of external requests in the $M/G/1/K$ queue is obtained based on Equation 6:

$$E(W_j) = \frac{1}{\Lambda_j} \sum_{k=0}^{K_j - 1} k \cdot P_{d,k}^j + \frac{K_j}{\Lambda_j}(P_{d,0}^j + \rho_e^j - 1) - E(T_j) \quad (6)$$

where, ρ_e^j is the queue utilization for external requests and is calculated based on Equations 4 as follows:

$$\rho_e^j = \Lambda_j \cdot E(T_j) = \frac{\omega \cdot \Lambda_j}{1 - \rho_l^j} \quad (7)$$

Also in Equation 6, $P_{d,k}^j$ is the probability that a newly arriving external request observes k requests waiting in the queue of cluster j. This is irrespective of whether or not the external request finally joins the queue. $P_{d,k}^j$ is obtained as follows [3]:

$$P_{d,k}^j = \frac{P_{\infty,k}^j}{\sum_{i=0}^{K_j - 1} P_{\infty,i}^j}, k = 0, 1, ..., K_j - 1 \quad (8)$$

Based on Equation 8, to work out $P_{d,0}^j$, we need $P_{\infty,0}^j$ and $P_{\infty,k}^j$. $P_{\infty,0}^j$ is equivalent to the probability of zero length queue in an $M/G/1$ queue, which is: $P_{\infty,0}^j = 1 - \rho_e^j$ [13]. However, $P_{\infty,k}^j$ is obtained according to Equation 9 [3].

$$P_{\infty,k}^j = \frac{1}{\mu_e^j} \cdot \left(a_{k-1} \cdot P_{\infty,0}^j + \sum_{i=1}^{k-1} a_{K_j - i} \cdot P_{\infty,i}^j\right) \quad (9)$$

where a_k^j is defined as follows:

$$a_k^j = \int_0^\infty \frac{(t\lambda_j)^k}{k!} \cdot e^{-t\lambda_j} \cdot b_j(t) \cdot d_t \quad (10)$$

$b_j(t)$ in Equation 10 is the probability density function (PDF) of service time for external requests in cluster j.

Gong et al. [7] showed that the service time of external requests in the presence of preemption in a cluster follows the Gamma distribution. Therefore, we can apply the moment matching to acquire the parameters of the Gamma distribution (*scale*(α) and *shape*(β)). In this case, $\alpha\beta = E(T_j)$ and $\alpha^2\beta = V(T_j)$ and consequently α and β are obtained as follows:

$$\alpha_j = \frac{\rho_l^j(\theta_j^2 + 1)}{\mu_e^j(1-\rho_l^j)^2}, \beta_j = \frac{(1-\rho_l^j)\mu_l^j \cdot \omega}{\rho_l^j(\theta_j^2 + 1)} \quad (11)$$

Hence, $b_j(t)$ in Equation 10 can be calculated as follows:

$$b_j(t) = \frac{(t/\alpha)^{\beta-1} \cdot e^{-t/\alpha}}{\alpha \cdot \Gamma(\beta)} \quad (12)$$

where $\Gamma(\beta)$ is the Gamma function.

A. The Proposed Admission Control Policy

Here, we discuss how the analysis mentioned in the previous section can be adapted as the admission control policy for external requests within each cluster of a Grid. The positioning of this policy is demonstrated as "*AC*" in Figure 2.

As we can see in Equation 1, solving the queuing model entails knowing "D" (deadline). The deadline of an external request includes the amount of time a request can possibly wait. For the sake of accuracy, similar to [6], [11], we have categorized external requests into *low-urgency* and *high-urgency* based on their criticality. In low-urgency requests the deadline is significantly greater than run time of the requests (i.e. $deadlineRatio = deadline/runtime$ is high). By contrast, in high-urgency requests the deadline ratio is low. Having the average deadline ratio for the low-urgency and high-urgency requests, the upper bound for response time of external requests in Equation 1 (D) is worked out as follows:

$$D = (rate_l \cdot u_l \cdot \omega) + ((1 - rate_l) \cdot u_h \cdot \omega) \quad (13)$$

where $rate_l$ determines the percentage of external requests which have low-urgency; also u_l and u_h are deadline ratios for low-urgency and high-urgency external requests, respectively.

Then, the preemption-aware admission control policy (PACP), which is built upon the analysis of Section IV, can

be constructed. This policy is shown in the form of pseudo-code in Algorithm 1. In the beginning of the Algorithm (step 1), D indicates the maximum time on average an external request can be postponed without violating its deadline. Next, in steps 4-10, in each iteration of the loop the queue length is increased by one up until the average response time ($E(R)$), in step 9, exceeds D.

Algorithm 1: Preemption-aware Admission Control Policy (PACP) in cluster j.

Input: Λ_j,θ_j,ω,λ_j,μ_e^j,μ_l^j,$rate_l$,u_l,u_h
Output: K_j (Queue length)

1 $D \leftarrow (rate_l * u_l * \omega) + ((1 - rate_l) * u_h * \omega)$;
2 $K_j \leftarrow 0$;
3 $ExpectedResponse_j \leftarrow 0$;
4 **while** $ExpectedResponse_j < D$ **do**
5 `/*calculating` $E(R)$ `for a queue with length` K_j `in cluster` j`*/`
6 $\sigma \leftarrow 0$;
7 **for** $N_q^j \leftarrow 0$ **to** $K_j - 1$ **do**
8 $\sigma += N_q^j \cdot P_{d,N_q^j}^j$;
9 $ExpectedResponse_j \leftarrow \frac{1}{\Lambda_j} \cdot \sigma_j + \frac{K_j}{\Lambda_j}(P_{d,0}^j + \rho_e^j - 1)$;
10 $K_j \leftarrow K_j + 1$;

The output of the algorithm is K_j, which is the ideal number of external requests that can be admitted by cluster j. Once K_j is obtained for cluster j, the LRM of the cluster will not accept any external requests beyond K_j.

It is worth mentioning that, in practice, the LRM in a cluster can get the required parameters for PACP by analyzing the clusters' workload. Such parameters have been used in similar researches [8], [24].

Although we considered the Poisson arrival in the analysis, in the next section we examine how efficient the provided analysis and the proposed policy would be under real arrival model (i.e. non-Poisson).

V. Performance Evaluation

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

A. Performance Metrics

1) Deadline Violation Rate (DVR): Measures the percentage of deadline violations within the whole external requests. In fact, users are interested in a policy that results in less deadline violations. High values of this metric express less user satisfaction. The DVR of external requests in a Grid is calculated based on Equation 14.

$$DVR = \frac{(a \cdot v) + r}{a + r} \cdot 100 \tag{14}$$

where a, r are the number of accepted and rejected external requests in a Grid. v is the deadline violation ratio within the accepted external requests ($0 \leq v \leq 1$).

2) Completed External Requests: Admission control policies usually limit the number of requests processed in a cluster. This is, however, against the resource owner's aim who benefits from processing as many requests as possible. Therefore, we are interested to see how different policies affect the number of completed external requests within each cluster and subsequently within a Grid.

B. Experimental Setup

We use GridSim [21], a discrete event simulator, to evaluate the performance of the admission control policy. We consider a Grid with 3 clusters with 64, 128, and 256 processing elements and with different computing speeds ($s_1 = 2000, s_2 = 3000, s_3 = 2100$ MIPS) respectively. These sizes are in accordance with the average demand of the current scientific high performance computing applications [9].

Each cluster is managed via an LRM with a conservative backfilling scheduler to improve the resource utilization [17]. We assume all processing elements of each cluster as a single core CPU with one VM. Since the requests contain moldable applications, the number of VMs required by a request adapts to the number of VMs in the allocated cluster and the duration (execution time) of the request changes accordingly.

The performance of our admission control policy also depends on the scheduling policy in the gateway (IGG) where incoming external requests are allocated to different clusters (see Figure 1). We applied several scheduling policies in the IGG to schedule the incoming external requests amongst different clusters of a Grid. However, because of lack of space we have reported the results where the round robin policy was applied in IGG. Based on this policy, the arrival rate of external requests to cluster j is: $\Lambda_j = \Lambda_g/n$ where Λ_g is the arrival rate to a Grid and n is the number clusters in the Grid.

1) Baseline Policies: For the sake of comparison, we evaluate PACP against 3 other policies. Details of these policies are described below:

- Conservative Admission Control Policy (CACP): As a baseline policy, this policy admits as many requests as assigned by the IGG. In fact, this policy favors resource owners since it does not reject any external request with the aim of maximizing the number of completed external requests. From the queuing model perspective, this policy considers an $M/G/1/\infty$ queue within each cluster, where queue length is infinite.
- Aggressive Admission Control Policy (AACP): The other baseline policy considers the other extreme of spectrum where each cluster accepts one external request at any time and tries to finish that within its deadline. We can argue that this policy favors accepted requests since it just tries to minimize deadline violation rate of accepted requests.
- Rate-based Admission Control Policy (RACP): Sharifian et al. [20] proposed this policy which is the most similar

to our policy in the area. In this policy the queue length is determined based on the service rate for external requests and local request arrival rate in a cluster (i.e. $N_q = \mu_e/\lambda$). We compare our proposed policy (PACP) with RACP to show its performance in comparison with recent works in the area.

2) Workload Model: In the experiments conducted, a workload model based on Grid Workload Archive (GWA) [10] has been configured to generate a two-day-long workload of bag-of-tasks requests. This model is based on traces of 7 Grids over a year and is a good representative for Grid workloads.

For the sake of accuracy, each experiment is carried out 10 times by using different workloads. The results of the experiments are analyzed from practical and statistical perspectives. In the statistical analysis we applied T-student test and we have verified the *normality* of the underlying data as well as equity of variance. Also, we assured that CV of all the reported results is less than 0.1. In Table II, different characteristics of the workload are described. Since the distribution of local requests and external requests are independent of each other in each cluster, we mention them in distinct columns. The values of parameters in Table II are chosen based on realistic values collected and analyzed by Iosup et al. [10].

In each experiment, we change one characteristic of the workload, while other characteristics are unchanged as follows:

- Arrival rate of local requests varies through changing α_l (we term it *local scale* which stands for scale parameter in the Weibull distribution). For external requests, we keep $M_e = 90$ seconds and α_e=1.7 (called *external scale*). For local requests we keep $M_l = 90$ seconds.
- Task duration of local requests varies: We keep $\alpha_e = 1.7$, $M_e = 90$ seconds, and $\alpha_l = 4$.
- Arrival rate of external requests varies: We keep $\alpha_l = 4$, $M_l = 90$ seconds, and $M_e = 90$ seconds.
- Task duration of external requests varies: We keep $\alpha_l = 4$, $M_l = 90$, and $\alpha_e = 1.7$.

There are also some points on the value of parameters in the workloads:

- In Table II, by increasing α_l and α_e inter-arrival time increases (i.e. requests arrive less often). Therefore, as we expect that in reality external requests arrive more frequently, we assign lower values of α to them.
- Mean duration of external tasks (ω), which is needed in algorithm 1, is $\omega = meanNumberofTasks * meanTaskDuration$.
- To be more realistic, the *local* workload assigned to each cluster is proportional to cluster capacity (i.e. bigger clusters are receiving more and bigger local requests). In fact, the values mentioned in Table II are the average characteristics of the *local* workload on the cluster with 128 processing elements. On the cluster with 64 processing elements, the mean task duration is decreased by 1 and the scale parameter (α_l) is increased by 1. In the cluster with 256 processing elements the mean task duration is increased by 1 and the α_l parameter is decreased by 1.

The GWA workload model does not have deadline for requests. Thus, similar to [6], [11], we synthetically assign deadlines to *low-urgency* and *high-urgency* external requests. Deadline ratio is distributed normally within each class of the requests. In our experiments, we consider the deadline ratio for low-urgency as: $N(4, 1)$ and for high-urgency external requests as: $N(2, 1)$. In fact, we have run the experiments for different values of deadline ratio in low and high-urgency requests. However, due to shortage of space we just report the mentioned situation. Finally, the arrival sequence of high-urgency and low-urgency request classes are distributed uniformly throughout the workload.

C. Experimental Results

1) Deadline Violation Rate (DVR): One of the goals of this paper is to express the impact of admission control policies on the deadline violation in a federated Grid and with the presence of preemption. Therefore, in this experiment we report the DVR for external requests when different admission control policies are applied.

As we can see in all sub-figures of Figure 3, PACP has resulted in less deadline violations when compared with other policies. Specifically, we observe a rise in DVR as the average duration of tasks in local and external requests increases (Figures 3(a) and 3(c)). In Figure 3(b) and 3(d) we notice that DVR in all policies decreases when the inter-arrival time of local requests and external requests increases. In fact, in Figure 3(b), when the inter-arrival time of local requests increases, fewer preemptions take place for external requests and thus DVR decreases. On the other hand, in Figure 3(d), as external requests become less frequent, fewer external requests join the queue and existing external request have more opportunity to get completed before their deadline.

In all cases, the difference between PACP and other policies is more significant when there is more load in the system which shows the efficiency of PACP when the system is heavily loaded. The only exception is when task duration for external requests is long (more than 100 seconds in Figure 3(c)) which indicates that when external requests are becoming long, the queuing policies cannot affect DVR significantly. In the best situation (in Figure 3(c), where the average task duration is 80 seconds) we observe that PACP results in around 20% less deadline violation when compared to RACP. In other points (between 70 and 90), the 95% confidence interval (CI) of the average difference between RACP and PACP is (14.79,18.56) where P-value$<$0.001.

This experiment also expresses that although AACP accepts few external requests, its DVR is the highest. This is because in Equation 14, the number of rejections is very high and, therefore, the value of r dominates the result. RACP in these experiments is functioning close to the CACP. Particularly, in Figure 3(d) since inter-arrival time of external requests increases, RACP admission rate increases and approaches CACP. However, in Figure 3(a) and 3(c)

TABLE II
PARAMETERS OF THE WORKLOAD MODEL.

Input Parameter	Distribution	external Requests	Local Requests
Inter-arrival Time	Weibull	$(0.2 \leq \alpha_e \leq 3.2, \beta_e = 7.86)$	$(2 \leq \alpha_l \leq 10, \beta_l = 7.86)$
No. of Tasks	Weibull	$(a_e = 1.76, b_e = 2.11)$	$(a_l = 1.76, b_l = 2.11)$
Task Duration	Normal	$(60 \leq M_e \leq 110, \sigma_e = 6.1)$	$(60 \leq M_l \leq 110, \sigma_l = 6.1)$

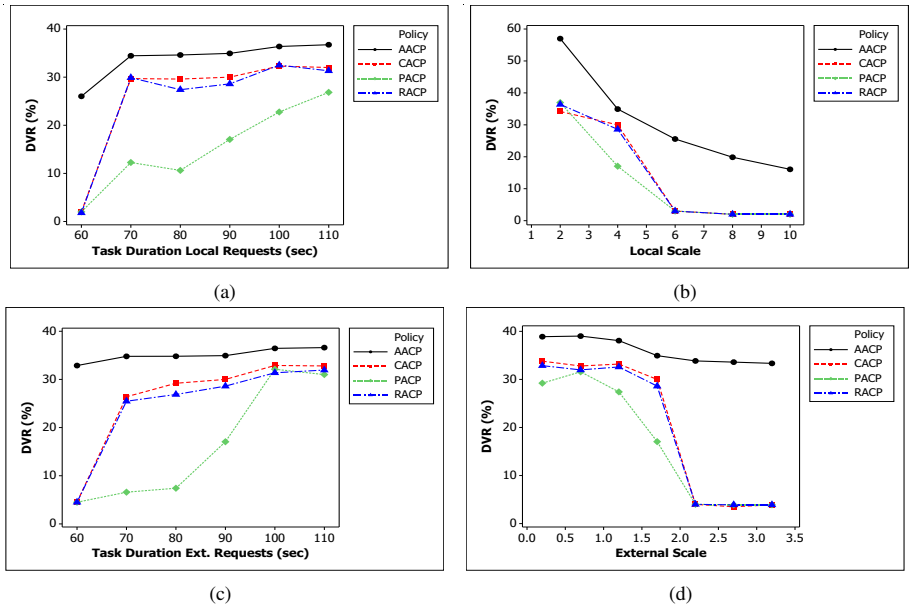

Fig. 3. Deadline Violation Rate (DVR) resulted from different policies. The experiment is carried out by modifying (a) the average local task duration (M_l) and (b) the scale parameter (α_l) in local requests. In (c),(d) for external requests with altering the mean task duration (M_e) and the scale parameter (α_e) as mentioned in Table II.

as service rate of external requests decreases, a greater difference appears between RACP and CACP. In Figure 3(a), the 95% CI of the average difference between RACP and PACP is (14.12,17.86) and P-value<0.001.

2) Completed External Requests: Resource owners, in general, prefer to run as many external requests as possible. Therefore, in this experiment we report the percentage of external requests getting served. Different sub-figures in Figure 4 show how various policies affect the number of completed external requests from different aspects.

In general, we observe in all sub-figures of Figure 4 that AACP leads to the least number of completed external requests (because of too many rejections) whereas CACP results in the most number of completed external requests (always 100%) because it does not reject any of the external requests. We also witness that PACP, almost in all cases outperforms RACP. Superiority of PACP is particularly remarkable when the local/external requests are not long. According to Figures 4(a) and 4(c), the percentage of completed external requests decreases by increasing the task duration for both local and external requests. Since PACP adaptively decreases the queue length, deadline violation rate is minimized. Hence, the percentage of completed external requests decreases. Similarly, in RACP by increasing task duration of local or external requests, service rate for external requests decreases and consequently, number of completed external requests reduces. However, these figures note that PACP still performs better in comparison with RACP. The result of the 95% CI of the average difference between RACP and PACP in Figure 4(a) is (14.12,17.86) and P-value<0.001; In the same figure, the most difference between the two policies is around 25% when task duration for local requests is 70-80 seconds. Moreover, the 95% CI of the average difference between RACP and PACP in Figure 4(c) is (17.09,21.3) and P-value<0.001.

Figure 4(d) shows that PACP leads to completion of more external requests in comparison with RACP. We notice that as the external requests become less frequent PACP and RACP approach CACP and finally when external scale parameter is more than 2.0 all the external requests are accepted.

VI. CONCLUSIONS AND FUTURE WORK

In this research we considered a virtualized federated Grid environment where local requests and external requests coexist in each cluster and local requests have preemptive priority over the external requests. In this environment, we explored the ideal number of external requests that a cluster can accept without violating deadlines. We developed a performance model based on queuing theory and then we proposed a preemption-aware admission control policy (PACP) in the LRM of each cluster which determines the ideal queue length for external requests in the cluster. We compared the performance of the proposed policy with 3 other policies. Results of the experiments indicate that the PACP significantly decreases the deadline violation rate for external requests (up to 20%). Additionally, PACP leads to completing more external requests (up to 25%) in a two-day-

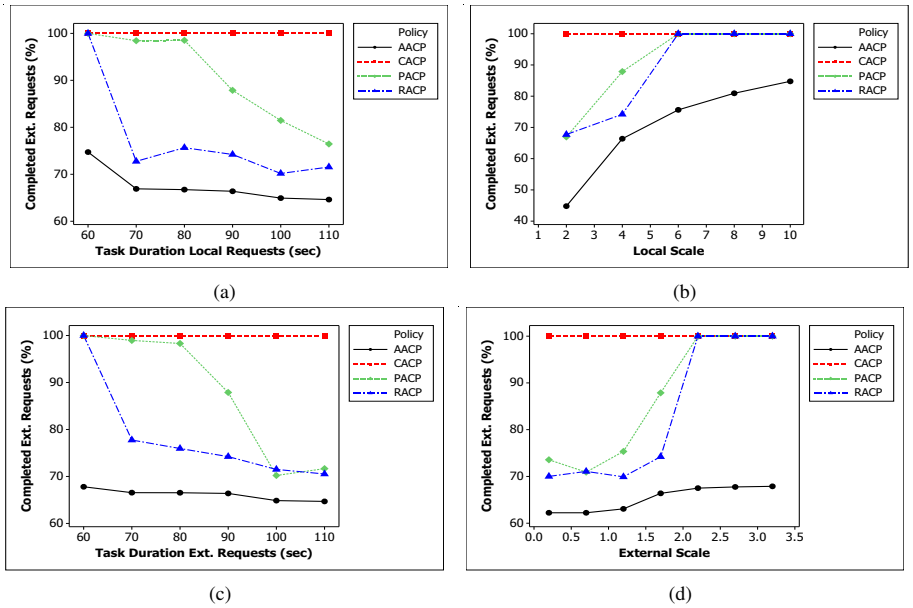

Fig. 4. Percentage of completed external requests resulted from different policies. The experiment is performed by modifying: (a) the mean task duration for local requests (M_l) and (b) the scale parameter (α_l) for local requests. Also, in (c) , (d) with altering the mean task duration for external requests (M_e); and scale parameter (α) of inter-arrival time for external requests.

long workload. In future, we plan to relax the assumption of moldable applications and solve the problem for all types of parallel requests. Additionally, we will consider revenue for resource owner based on different types of requests.

REFERENCES

[1] J. Almeida, V. Almeida, D. Ardagna, i. Cunha, C. Francalanci, and M. Trubian. Joint admission control and resource allocation in virtualized servers. *Jnl. Parallel and Distributed Computing*, 70:344–362, April 2010.

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

[3] S. Bose. *An introduction to queueing systems*. Springer US, 2001.

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

[5] J. Fontán, T. Vázquez, L. Gonzalez, R. S. Montero, and 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, May 2008.

[6] S. K. Garg, C. S. Yeo, A. Anandasivam, and R. Buyya. Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers. *Jnl. Parallel and Distributed Computing*, 71(6):732 – 749, 2011.

[7] L. Gong, X.-H. Sun, and E. Watson. Performance modeling and prediction of nondedicated network computing. *IEEE Transactions on Computers,*, 51(9):1041 – 1055, sep 2002.

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

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

[10] A. Iosup, O. Sonmez, S. Anoep, and D. Epema. The performance of bags-of-tasks in large-scale distributed systems. In *Proceedings of the 17th international symposium on High performance distributed computing*, pages 97–108, USA, 2008.

[11] D. Irwin, L. Grit, and J. Chase. Balancing risk and reward in a market-based task service. In *13th IEEE International Symposium on High performance Distributed Computing,*, pages 160 – 169, june 2004.

[12] M. Islam, P. Balaji, P. Sadayappan, and D. Panda. Qops: A QoS based scheme for parallel job scheduling. In *Job Scheduling Strategies for Parallel Processing*, pages 252–268. Springer, 2003.

[13] L. Kleinrock. *Queueing systems: Computer applications*. Wiley-interscience, 1976.

[14] R. Kubert and S. Wesner. Service level agreements for job control in high-performance computing. In *International Multiconference on Computer Science and Information Technology,*, pages 655 –661, oct. 2010.

[15] J. Liang and M. Nahrstedt. Supporting quality of service in a non-dedicated opportunistic environment. *IEEE International Symposium on Cluster Computing and the Grid*, pages 74–81, 2004.

[16] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The Eucalyptus open-source cloud-computing system. In *1st Workshop of Cloud Computing and Its Applications*, October 2008.

[17] G. Sabin, R. Kettimuthu, A. Rajan, and P. Sadayappan. Scheduling of parallel jobs in a heterogeneous multi-site environement. In *JSSPP*, pages 87–104, 2003.

[18] T. Sandholm, K. Lai, and S. Clearwater. Admission control in a computational market. In *Eighth IEEE International Symposium on Cluster Computing and the Grid*, pages 277–286. IEEE, 2008.

[19] R. Schneider, G. Faust, U. Hindenlang, and P. Helwig. Inhomogeneous, orthotropic material model for the cortical structure of long bones modelled on the basis of clinical ct or density data. *Computer Methods in Applied Mechanics and Engineering*, 198(27-29):2167 – 2174, 2009.

[20] S. Sharifian, S. Motamedi, and M. Akbari. A content-based load balancing algorithm with admission control for cluster web servers. *Future Generation Computer Systems*, 24(8):775–787, 2008.

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

[22] J. L. Vázquez-Poletti, E. Huedo, R. S. Montero, and I. M. Llorente. A comparison between two grid scheduling philosophies: EGEE WMS and grid way. *Multiagent Grid Syst.*, 3:429–439, December 2007.

[23] P. Xavier, W. Cai, and B.-S. Lee. A dynamic admission control scheme to manage contention on shared computing resources. *Concurrency and Computing: Practice and Experience*, 21:133–158, February 2009.

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