Information and Software Technology 51 (2009) 42–55

Contents lists available at ScienceDirect

Information and Software Technology

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

Performance analysis of allocation policies for interGrid resource provisioning

Marcos Dias de Assunção [a,b,*], Rajkumar Buyya [a]

[a] Grid Computing and Distributed Systems (GRIDS) Laboratory, Department of Computer Science and Software Engineering, 111 Barry Street, The University of Melbourne, Carlton, Victoria 3053, Australia
[b] NICTA Victoria Research Laboratory, The University of Melbourne, Victoria 3010, Australia

ARTICLE INFO

Keywords:
Resource provisioning
Grid computing
InterGrid resource allocation

ABSTRACT

Several Grids have been established and used for varying science applications during the last years. Most of these Grids, however, work in isolation and with different utilisation levels. Previous work has introduced an architecture and a mechanism to enable resource sharing amongst Grids. It has demonstrated that there can be benefits for a Grid to offload requests or provide spare resources to another Grid. In this work, we address the problem of resource provisioning to Grid applications in multiple-Grid environments. The provisioning is carried out based on availability information obtained from queueing-based resource management systems deployed at the provider sites which are the participants of the Grids. We evaluate the performance of different allocation policies. In contrast to existing work on load sharing across Grids, the policies described here take into account the local load of resource providers, imprecise availability information and the compensation of providers for the resources offered to the Grid. In addition, we evaluate these policies along with a mechanism that allows resource sharing amongst Grids. Experimental results obtained through simulation show that the mechanism and policies are effective in redirecting requests thus improving the applications' average weighted response time.

1. Introduction

Advances in Grid computing have enabled the creation of Grid-based resource sharing networks such as TeraGrid [1,2], Naregi [3], Open Science Grid [4], and PlanetLab [5]. These networks, composed of multiple resource providers, enable collaborative work and resource sharing amongst groups of individuals and organisations. These collaborations, widely known as virtual organisations (VOs) [6], require resources from multiple computing sites.

Although these Grids have contributed to various sciences and disciplines, they mostly work in isolation. The Grid interoperability now–community group (GIN–CG) [7] is working on providing interoperability between Grids by developing components and adapters that enable secure job submissions, data transfers and information queries. Even though GIN–CG's efforts are relevant, its members also highlight the need for common allocation and brokering of resources across Grids.[1] In addition, Iosup et al. [8] have identified the need for resource management across Grids. They have shown that there is a load imbalance between current Grids. Moreover, interoperability and common protocols are important, but not enough to interconnect Grids. For example, a set of common communication protocols underly the Internet, but when internet service providers (ISPs) peer with one another they consider their policies, economic issues, and the social and economic impact of peering [9,10].

The resource utilisation within a Grid has fixed and operational costs such as those with electricity providers and system administrators. Consequently, when Grids are connected with one another, it is relevant to take into account these costs. There can be benefits for a Grid to provide spare capacity to peering Grids, possibly in return for regular payments, and to acquire resources from peering Grids to serve occasional internal peak demands. We have proposed in previous work [11] a resource exchange mechanism that enables a Grid, under peak load conditions, to redirect requests to another Grid. In contrast to existing work in load management across Grids [8,7] the proposed mechanism takes into account the compensation of resource providers for the resources offered to the Grid; this compensation corresponds to the amount paid by a broker for resources utilised.

In this work, we address the problem of resource provisioning in environments with multiple Grids. Emerging deadline-driven Grid applications require access to several resources and predict-

* Corresponding author. Address: Grid Computing and Distributed Systems (GRIDS) Laboratory, Department of Computer Science and Software Engineering, 111 Barry Street, The University of Melbourne, Carlton, Victoria 3053, Australia. Tel./fax: +61 43 7048238.
E-mail addresses: marcosd@csse.unimelb.edu.au (M.D. de Assunção), raj@csse.unimelb.edu.au (R. Buyya).

[1] The personal communication amongst GIN–CG members is online at: http://www.ogf.org/pipermail/gin-ops/2007-July/000142.html.

0950-5849/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2008.09.013

able quality of service (QoS). However, it is difficult to provision resources to these applications because of the complexity of providing guarantees about the start or completion times of applications currently in execution or waiting in the resources' queues. The resources contributed by providers to a Grid are generally clusters of computers managed by queueing-based resource management systems (RMSs), such as PBS [12] and Condor [13]. These RMSs generally use optimisations to the first come first served (FCFS) policy such as backfilling [14] in order to reduce the scheduling queue's fragmentation, improve job response time and maximise resource utilisation. These optimisations make it difficult to predict the resource availability over a time frame as the jobs' start and completion times are dependent on resource workloads.

To complicate matters further, Grid users commonly access resources from a Grid via mediators such as brokers or gateways [15,16]. The design of gateways that provision resources to deadline-driven applications relying on information given by current RMSs may be complex and prone to scheduling decisions that are far from optimal. Furthermore, a gateway representing a Grid can have peering arrangements or contracts with other gateways through which they co-ordinate the resource provisioning. This complicates provisioning as a gateway needs not only to provision resources to its users, but also provision spare capacity to other gateways. Previous work has demonstrated how information about fragments in the scheduling queue of clusters, or free time slots, can be obtained from RMSs and provided to gateways to be provisioned to Grid applications [17,18].

In previous work we have evaluated the precision of the availability information obtained via different techniques and their impact on provisioning in multiple site environments [18]. In the present work, we utilise this information as the basis for resource provisioning across Grids. We extend previous studies on InterGrid load management [11] and utilise the resource provisioning techniques used for multiple site environments as follows. Firstly, we extend the provisioning policies in order to consider the cost of delegating resources to a gateway. Second, the mechanism for load management across Grids that previously assumed an ON/OFF approach for modelling resource providers' load, now utilises information obtained from scheduling policies using conservative backfilling [19] and multiple resource partitions [20]. Previous studies have used ON/OFF models [21] wherein ON and OFF intervals represent off-peak and peak periods, respectively. However, the queuing-based scheduling policies enhance the evaluation of the overall mechanism by modelling a scenario closer to reality. Third, the experiments carried out in this work measure the jobs' average weighted response time for each scenario. Moreover, we evaluate whether the proposed policies have a smaller impact on local providers' jobs than traditional techniques when the Grids are interconnected.

The rest of this paper is organised as follows. Section 2 presents the related work. In Section 3, we describe the InterGrid scenario. We describe the policies used by resource providers in Section 4. Section 5 discusses the resource provisioning and load sharing across Grids. We present and elaborate on the performance evaluation and experimental results in Section 6. Section 7 concludes the paper and presents future work.

2. Related work

This section is organised as follows. Section 2.1 describes mechanisms and optimisations for scheduling and provisioning resources at a site level. Section 2.2 analyses previous systems that attempt to enable load sharing in multiple site environments and amongst resource sharing networks.

2.1. Resource provisioning at a site level

2.1.1. Modelling providers' resource availability

AuYoung et al. [21] consider a scenario wherein service providers establish contracts with resource providers. The availability information is modelled as ON/OFF intervals, which correspond to off-peak and peak periods, respectively. However, they do not demonstrate in practice how this information can be obtained from RMSs.

2.1.2. Advance reservations and creation of alternatives to rejected requests

Mechanisms for elastic advance reservations and generation of alternative time slots for advance reservation requests have been proposed [22,23]. These models can be incorporated in the provisioning scenario described in this work to improve resource utilisation and generate alternative offers for provisioning violations. However, we aim to reduce the interaction between resource providers and gateways by allowing the providers to inform the gateways about their spare capacity. We focus on how the availability information can be obtained from RMSs and how reliable it is under different conditions.

2.1.3. Resource allocation in consolidated centres

Padala et al. [24] apply control theory to address the provision of resources to multi-tier applications in a consolidated data centre. Garbacki and Naik [25] consider a scenario wherein customised services are deployed on virtual machines which in turn are placed into physical hosts. Although the provisioning of resources to applications in utility data centres is important, here we focus on traditional queue-based RMSs.

2.1.4. Resource provisioning

Singh et al. [17,26] have presented a provisioning model wherein Grid sites provide information on the time slots over which sets of resources will be available. The sites provide their resources to the Grid in return for payments, thus they present a cost structure consisting of fixed and variable costs over the resources provided. The provisioning model is evaluated considering the scheduling of workflow applications. The main goal is to find a subset of the aggregated resource availability, termed resource plan, such that both the allocation cost and the application make span are minimised. They utilise a multi-objective genetic algorithm (MOGA) approach to approximate the group of resource plans that correspond to the *Pareto-optimal* set. Experiments have been carried out considering one cluster and one broker at a time. Our work is different in the sense that we investigate multiple approaches to obtain availability information and how reliable this information can be in multiple site environments [18]. In this paper we utilise this availability information and evaluate the load sharing across brokers (in this work termed gateways), which has not been explored by Singh et al.

2.1.5. EASY backfilling and conservative backfilling

Schedulers generally use optimisations to the FCFS policy such as EASY backfilling (also known as aggressive backfilling) and conservative backfilling [19]. Backfilling allows a job to jump in the queue and execute earlier than jobs that arrived before it, given that enough resources are available and other waiting jobs are not delayed. Under conservative backfilling, a job can be used to backfill and execute earlier if it does not delay any other job waiting in the queue. EASY backfilling, on the other hand, uses a job to backfill and start execution if it does not delay only the first job in the queue – also termed pivot job. Under EASY backfilling, the schedule generally contains the expected completion of running jobs and the start time of the pivot job only. Some schedulers allow

the system administrator to configure the maximum number of pivots, which in turn enables the scheduler to maintain the start and expected completion times of up to the maximum number of pivot jobs [27]. In such case, if the maximum number of pivots is set to 5, for example, and there are 5 jobs waiting in the queue, a 6th job that just arrived is used to backfill if it does not delay any of the 5 pivot jobs. If the maximum number of pivots is set to a large number, the EASY backfilling algorithm becomes conservative backfilling. We utilise and extend policies based on these techniques for resource provisioning.

2.1.6. Multiple resource partition policies

Work on multiple resource partitions and priority scheduling has shown to reduce the job slowdown compared to EASY backfilling policies [20]. We build on this effort and extend it to enable other multiple partition policies. We also propose a new multiple resource partition policy based on load forecasts for resource provisioning.

2.2. Load sharing amongst resource sharing networks

2.2.1. Resource sharing networks and inter-operation efforts

Several Grid-based resource sharing networks have been built over past few years [1,3,4,28–31]. Recently, initiatives have emerged for linking resource sharing networks [32,33,7]. OneLab2 [32] and the Global Environment for Network Innovations (GENI) [33] have evolved from the PlanetLab architecture [5] to allow the federation of autonomous networks controlled by different organisations. Other Grid middleware interoperability approaches have also been presented [34]. These efforts provide the basis for load management across Grids by facilitating standard job submission and request redirection. We attempt to build on these efforts to investigate resource management across Grids.

2.2.2. Intermediate resource agents

Shirako [35,36] is a resource management system based on a resource leasing abstraction. Sites delegate limited power to allocate their resources by registering their offerings with brokers. Guest applications can acquire resources from brokers by leasing them for a specified time. Grit et al. [37] investigate the number of virtual machine (VM) migrations incurred when a broker and provider sites use either conflicting or synchronised policies for resource provisioning and VM placement. They show that when providers and the broker use conflicting policies, the number of migrations can be high. The present work does not investigate VM migrations and models. Additionally, while Grit et al. investigate the impact of conflicting policies, we take into account the resource cost and the exchange of resources by brokers.

2.2.3. Federated clusters and load sharing

Ranjan et al. [38] have proposed a service level agreement (SLA) based coordination mechanism for Grid superscheduling. A Grid Federation Agent (GFA) is the resource management system responsible for scheduling jobs at a cluster level. A GFA negotiates contracts and redirects requests to another cluster through a contract net based protocol. In the present work, an InterGrid Gateway is a broker with information about the free time slots at a group of resource providers. In addition, in contrast to GFAs, gateways do not engage into bilateral negotiations if the requests can be handled locally without increasing the cost above its threshold.

Iosup et al. [8] introduce a matchmaking protocol in which a site binds resources from remote sites to the local environment. Delegated matchmaking enables requests for resources to be delegated up and down the proposed site hierarchy. Our model shares aspects with Iosup et al.'s work, in the sense that InterGrid Gateways work as site recommenders so matching requests to resources available. However, it differs with respect to the resource exchange protocol and the consideration for compensation of resource providers for the resources acquired by a gateway.

The mechanism presented in this work derives from Medusa [39], a stream processing system that allows the migration of stream processing operators from overloaded resources to resources with spare capacity. However, it differs in terms of the negotiation protocol for exchanging resources between Grids and the resource selection and request redirection policies. The resource selection policies take into account the availability information from multiple sites, which in turn use scheduling optimisations such as backfilling. The negotiation protocol considers one round of counter-offers whilst the resource price is computed with the availability information sent by providers.

Wang and Morris [40] provide a taxonomy on load sharing in distributed systems. Some findings include that efficiency in load sharing depends on the environment and server-initiative tends to outperform source-initiative strategies when the same amount of information about stakeholders is available. In our scenario, resources have multiple processors; also, resources are heterogeneous in the number of processors. These conditions make the local scheduling subproblem different; in addition, resource management across Grids introduces a third subproblem: the load sharing between Grids. Surana et al. [41] address the load balancing in DHT-based P2P networks. Nodes of the P2P system run virtual servers responsible for ranges of objects in the address space; they inform directories about the load in the virtual servers whereas the directories periodically compute reassignments and trigger migrations of virtual servers to achieve balance. The present work, in contrast, does not perform migration of virtual servers, and focuses on the redirection and assignment of resources to requests.

2.2.4. Economics inspired resource allocation

A number of approaches have been proposed which use economic models to address resource usage and incentives in a Grid [42–47]. Particularly, a well-designed market-based resource allocation mechanism provides incentives for participation by ensuring that all the actors in the system maximise their utility and do not have incentives to deviate from the designed protocol [48]. We use a simple economic inspired mechanism in which gateways representing Grids redirect or accept requests based on the cost of serving them. However, we focus on the evaluation of system performance metrics such as response time of both Grid and providers' local jobs.

The negotiation protocol used among gateways has been described by Rosenschein and Zlotkin [49] as a monotonic concession protocol. In our implementation, there is one round of concessions in which the gateway that receives an offer to process a request creates a counter-offer if a deal cannot be initially reached.

3. Provisioning in InterGrid environments

Our previous work has introduced an architecture, called the InterGrid, based on gateways that mediate resource exchange between Grids. It allows participants to allocate resources from different Grids in a seamless manner. We provide an overview of the InterGrid in this section, but for more details about the architecture and project goals we refer to previous work [50].

The InterGrid is inspired by the peering agreements between ISPs. The Internet is composed of competing ISPs that agree to allow traffic into one another's networks. These agreements between ISPs are commonly termed as peering and transit arrangements

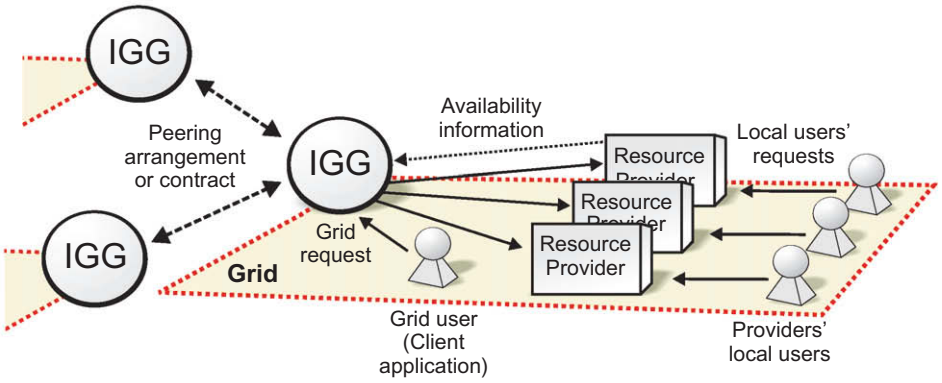

Fig. 1. The provisioning scenario with multiple Grids considered in this work.

[9]. Fig. 1 provides an overview of the InterGrid architecture and the resource provisioning scenario considered in this work.

A resource provider (RP) contributes a share of computational resources, storage resources, networks, application services or other type of resource to a Grid in return for regular payments. An RP has local users whose resource demands need to be satisfied, yet it delegates provisioning rights over spare resources to an InterGrid gateway (IGG) by providing information about the resources available in the form of free time slots. A free time slot includes information about the number of resources available, their configuration and time frame over which they will be available. Section 4 describes how this information can be obtained from resource management systems. The resources provided can be physical or virtual resources such as VMs and the delegation can be made through a secure protocol such as SHARP [51]. Protocols for secure delegation, however, are not in the scope of this paper; this work focuses on the resource provisioning aspect. Although a Grid can have a resource management system of its own (i.e. an IntraGrid resource manager), for the sake of simplicity, here an RP delegates provisioning rights directly to an IGG.

A Grid has pre-defined peering arrangements with other Grids, managed by IGGs and, through which they co-ordinate the use of resources of the InterGrid. An IGG is aware of the terms of the peering with other Grids; provides Grid selection capabilities by selecting a suitable Grid able to provide the required resources; and replies to requests from other IGGs. The peering arrangement between two Grids is represented as a contract. Request redirection policies determine which peering Grid is selected to process a request and at what price the processing is performed. The assumption of pre-established arrangements is reasonable as current Grids need to reserve the network links and set up the resources required.[2]

When a Grid user needs to deploy or execute an application, he/she requests the IGG for a number of resources. When the individual Grid cannot provide the required resources, the IGG selects a peering Grid based on the agreements and the policies in place. The user is then given a resource ticket granting access to the resources, which will later be passed to the selected provider in return for the required resources.

A request corresponds to an individual job whereas an application can comprise several jobs. A request is contiguous and needs to be served with resources from a single resource provider. Allowing a request to be served with resources from multiple resource providers may require co-allocation which is not addressed in this work. A request received by an IGG contains a description of the required resources and the usage time. The request can require a best effort service, meaning that the resources can be provided at any time as long as they are made available for the requested usage time. Alternatively, for some requests, the resources need to be allocated within a specified deadline.

3.1. The resource exchange

The peering agreements between Grids define (i) what resources are exchanged between the Grids and the price of the resources exchanged. The policies specify when an IGG redirects requests to another, and when a request redirected by one IGG is accepted by another IGG. Therefore, the goal of a participating IGG is to (i) serve its user communities by providing allocations that assign resources to satisfy their requirements; (ii) offer spare resources to peering Grids under some compensation; and (iii) acquire resources from other Grids to satisfy its users under peak load conditions.

4. Resource provider policies

We have previously investigated resource providers' scheduling policies which enable an IGG to obtain resource availability information in the form of free time slots [18]. This work considers a subset of the investigated policies, which have previously demonstrated good performance. The policies utilise an 'availability profile' similar to that described by Mu'alem and Feitelson [19]. The availability profile is a list whose entries describe the CPU availability at particular times in the future. These correspond to the completion or start of jobs and advance reservations. A job A whose start time or completion time coincides with either the start or completion of another job B, may share entries with B in the profile. By scanning the availability profile and using load forecasts the resource providers inform the gateway about the free time slots; the gateway in turn can carry out provisioning decisions based on this information.

4.1. Conservative backfilling based policies

Under conservative backfilling, a job is used to backfill and start execution earlier than expected, given that it does not delay any other job in the scheduling queue [19]. In order to reduce complexity, the schedule for the job is generally determined at its arrival and the availability profile is updated accordingly. Given those conditions, it is possible to obtain the free time slots by scanning the availability profile. This approach, depicted in Fig. 2a, was also used by Singh et al. [17,26]. In that case, the availability profile is scanned until a given time horizon thus creating windows of availability or free time slots; the finish time of a free time slot is either

[2] An example includes the interconnection of Grid'5000 with DAS-3 and NAREGI. More information is available at http://www.grid5000.fr.

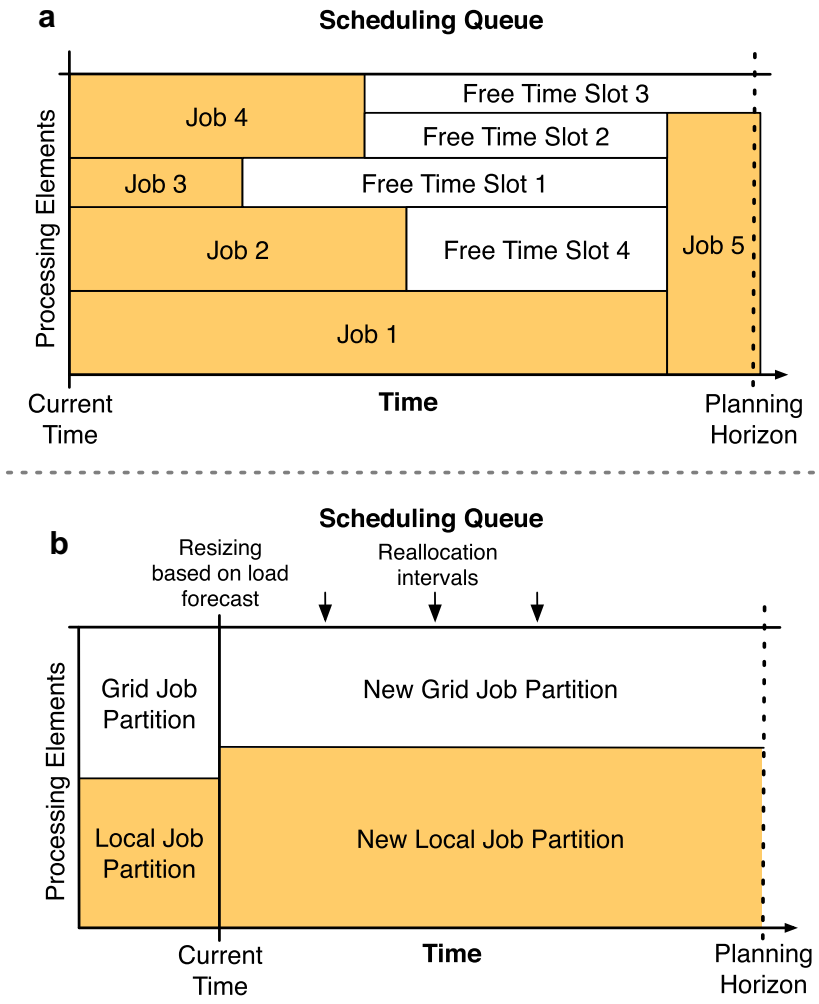

Fig. 2. Obtaining the free time slots: (a) by using conservative backfilling and (b) using multiple resource partitions.

the finish time of a job in the waiting queue or the planning horizon. If the horizon is set to ∞, the provider will be disclosing all the information. The availability information can either be provided on a periodical basis wherein provider and gateway agree on an interval at which the former sends the availability information or upon the gateway's request.

4.2. Multiple resource partition policies

In previous work we have evaluated three multiple resource partition policies [18]. In the present work we describe only two policies that have demonstrated good performance. The multiple resource partition policies divide the resources available in multiple partitions and assign jobs to these partitions according to predicates. A partition can borrow resources from another when they are not in use by the latter and borrowing is allowed by the scheduler. The policies are based on the idea of multiple resource partitions described by Lawson and Smirni [20]. Lawson and Smirni's original policy implements EASY backfilling. In this case, each partition uses EASY backfilling with one pivot, which is the first job in the waiting queue for that partition. A job belonging to a given partition can start its execution if it does not delay the partition's pivot and the partition has enough resources. In case the partition does not have enough resources, the job can still start execution if additional resources can be borrowed from other partitions without delaying their pivots. Additionally, the policy uses priority scheduling wherein the waiting queue is ordered by priority when the scheduler is backfilling.

One of the policies we have introduced, depicted in Fig. 2b, can alternate between EASY backfilling and conservative backfilling. The policy starts an interval with EASY backfilling, at which the partitions are resized by the scheduler based on a load forecast computed from information collected at previous intervals. As load forecasts are prone to be imprecise, when the scheduler resizes partitions, it also schedules reallocation events. At a reallocation event, the scheduler evaluates whether the load forecast has turned out to be an underestimation or not. If the load was underestimated, the policy resizes the partitions according to the load from the last resizing period until the current time and backfill the jobs, starting with the local jobs. We use EASY backfilling with configurable maximum number of pivots, similarly to MAUI scheduler [27]. The policy can be converted to conservative backfilling by setting the number of pivots to a large value, here represented by ∞. The other multiple resource partition policy also starts with two partitions, but uses only conservative backfilling with no load forecast technique.

Algorithm 1 describes two procedures used by the load forecast policy. The policy invokes *getFreeTimeSlots* every time the provider needs to send the availability information to the gateway; procedure *getFreeTimeSlots* schedules a later call for *reallocationEvent* to verify whether the previous forecast has turned out to be precise or if a reallocation is required.

Algorithm 1: Provider's load forecasting policy.

```
1  procedure getFreeTimeSlots()
2  begin
3      set the number of pivots of local and Grid partitions to ∞
4      schedule / back- ll jobs in the waiting queue
5      set the number of pivots of local and Grid partitions to 1
6      actualLoad ← load of waiting/running jobs
7      forecast ← get the load forecast
8      percToProvide ← min {0,1 ←actualLoad}
9      slots ← obtain the free time slots
10     slots ← resize slots according to percToProvide
11     if percToProvide > 0 then
12         inform gateway about slots
13         schedule reallocation event
14     schedule next event to obtain free time slots
15 end
16 procedure reallocationEvent()
17 begin
18     localLoad ← obtain the local load
19     forecast ← get the previously computed forecast
20     if localLoad > forecast then
21         set the number of pivots of local partition to ∞
22         schedule / back- ll jobs in the waiting queue
23         set the number of pivots of Grid partition to ∞
24         schedule / back- ll jobs in the waiting queue
25         slots ← obtain the free time slots
26         inform gateway about slots
27     else
28         schedule the next reallocation event
29 end
```

Line 3 and 4 of Algorithm 1 change the scheduler's backfilling strategy to conservative by setting the number of pivots in each partition to ∞. They also schedule the jobs currently waiting in the queue. After that, the scheduler's backfilling strategy is returned to EASY (line 5). Then, from line 6 to 10, the scheduler obtains the load forecast and the free time slots and resizes the free time slots by modifying the number of CPUs according to the number of resources expected to be available over the next interval. Next, the scheduler triggers a reallocation event. From line 20 to 24 the scheduler verifies whether the forecast was underestimated. If that is the case, the scheduler gives in and turns its backfilling strategy back to conservative and informs the gateway about the availability.

In previous work the resource providers' have not considered the pricing of resources [18]. In this work we apply a mechanism for pricing the free time slots delegated by a resource provider to the gateway. The resource providers use a pricing function for a resource unit given by Eq. 1.

$$price = cost + (cost * load) \quad (1)$$

where *cost* is the fixed cost of a unit of resource at the provider; *load* is obtained from the policy in use by the resource provider; *load* is either the load estimate when the policy supports forecasts or the actual load of both running and waiting jobs in the queue when forecasts are not used. A resource unit corresponds to one resource per second (i.e. a second of a CPU). Although straightforward, this pricing function has two components that capture namely the fixed cost of resources and the variable price caused by the demand.

5. Resource provisioning and load sharing

The adoption of economic principles for load sharing amongst Grids comes from observing economic institutions in the real world and how they regulate the allocation of resources, goods and the use of services [44]. Economic approaches are useful for coping with problems like providing Grid resources to different users with diverging QoS requirements and how to reward resource suppliers. As described beforehand, we envision that the interconnection of Grids involves problems that are similar to those faced by the peering of ISPs in the Internet. In the Internet, ISPs are in the business to make a profit – they see one another as competitors or sources of revenue – but they peer for economic or technical reasons [10,52–54]. ISPs have fixed and variable costs with network infrastructure, yet interconnect their network domains allowing traffic into one another's networks because they can benefit from peering by having a larger coverage or by offloading expensive links. Similarly, the use of Grid resources has fixed and variable costs and Grids may benefit from peering with one another. This section describes a mechanism for request redirection amongst Grids that takes into account their cost.

For each IGG_i, the allocation of its resources by its user communities over a unit of time represents a cost. The real-valued total cost function of IGG_i is represented by $cost_i(L)$, where $0 \leqslant L \leqslant 1$ is the current load determined by the number of resource units available in its Grid. For simplicity, here a resource unit corresponds to one resource per second (i.e. a second of a CPU). Therefore, the total cost given by $cost_i(L)$ depends on the number of resources allocated by the requests. Although each Grid could have its own cost function, in this work, the participating Grids utilise a quadratic cost

function. Such a function reflects the case of a network infrastructure in which it is highly costly to keep the resources operating at full capacity, mainly because it does not offer any provision for handling peak demands. This way, we believe that a function with a steep cost when the system approaches full utilisation reflects the case of current computing and network infrastructure. Moreover, a non-linear function is required by Eq. 4 to specify contracts with price-ranges as discussed later in this section.

The cost function $cost_i(L)$ is given by $[L_{units} * (p_{cost} + (p_{cost} * (\beta L)^2))]$, where L_{units} is the number of units in use at load L, β is a small constant value that determines how steep the cost curve is as the load approaches 1 and p_{cost} is the average price that IGG_i pays to resource providers for a resource unit. The price of a resource unit within IGG_i is given by the second part of the cost function (i.e. $p_{cost} + (p_{cost} * (\beta L)^2)$. We derive the average price p_{cost} paid by IGG_i to resource providers for a resource unit using by the following equation:

$$p_{cost} = \sum_{i=1}^{n} \left(cp_i \left(\frac{ru_i}{\sum_{j=1}^{n} ru_j} \right) \right) \quad (2)$$

where n is the number of resource providers in IGG_i's Grid; cp_i is the price of a resource unit at resource provider i; and ru_i is the number of resource units contributed by provider i until a given time horizon. This horizon is the request deadline when calculating the request cost described next. When updating the prices for resource units specified in the contracts, the horizon is the time of the next contract update (i.e. the next time when the IGGs update the prices of units negotiated). This way, L is dependent on how many resource units are available from the start time until the horizon and how many units are in use.

A request redirection is decided based on the per-request cost $mc_i : (u, L) \rightarrow \Re$ which is the increment in total cost for IGG_i for agreeing to provide resource units required by request u given its current load or allocations. If request u requires resource units that place u_{load} load in IGG_i's Grid, then the cost of serving u is derived by Eq. 3. If request u requires one resource unit, then the request cost is equals to a *unit cost*.

$$mc_i = cost_i(L + u_{load}) - cost_i(L) \quad (3)$$

IGG_i has a load threshold, by crossing which IGG_i considers itself overloaded. The redirection of requests is enabled between Grids that have negotiated contracts, at within the contracted price-range. A contract $C_{i,j}$ between IGG_i and IGG_j has a price-range $PR(C_{i,j}) : [price_{min}, price_{max}]$, where $price_{min}$ and $price_{max}$ are the minimum and maximum prices, respectively paid by IGG_i for a resource unit allocated from IGG_j. IGG_i can have contracts with multiple Grids. During periods of peak load, IGG_i can redirect requests to IGG_j if and only if both have a contract. Based on the current load levels, they agree on a final price $price_{final}$ within $PR(C_{i,j})$. IGG_i pays the amount equivalent to ($price_{final} *$ *number of units*). The redirection occurs when a Grid forwards requests to another because the cost of fulfilling the requests is higher than the amount that it would have to pay to the other Grid to serve them.

5.1. Contract types

We support two kinds of contracts: fixed-price and price-range contracts. A fixed-price contract is given by $PR(C_{i,j}) : [price_{max}, price_{max}]$ where $price_{max}$ is the fixed-price and a price-range contract corresponds to $PR(C_{i,j}) : [price_{max} - \Delta, price_{max}]$, where Δ determines the price-range. In the case of price-range contracts, participating Grids have to negotiate the final price at runtime. As discussed by Balazinska et al. [39], a load management mechanism based on fixed-price contracts may present disadvantages in some cases. For example, it reduces the flexibility in redirecting requests as a Grid can only offload requests if their cost is higher than the exact price it would pay to another Grid (i.e the number of resource units required by the request multiplied by the unit cost specified in the contract).

We define the price-range for a resource unit considering the decrease of load k from the load L. Let u be a request that requires u_{units} resource units and causes an increase in load u_{load}. The decrease in the per-unit cost due to removing k from the Grid's L is represented by δ_k, which is defined by the following equation.

$$\delta_k(L) = \frac{mc(u, L - u_{load}) - mc(u, L - k - u_{load})}{u_{units}} \quad (4)$$

δ_k is the approximate difference in the cost function gradient evaluated at the load level including and excluding load k. Given a contract with fixed-price $price_{max}$, L is the maximum load that an IGG can approach before its per resource unit cost exceeds $price_{max}$. In order to estimate the price-range for a resource unit in the contracts in our experiments, we let L be the load threshold; u_{units} be 1 and $\Delta = \delta_k$. We evaluate different values for L and k.

5.2. Provisioning policies

The policies described in this section define how an IGG offloads requests to peering Grids considering a contract network and how it accepts requests from other Grids.

During a time interval, IGG_i stores the requests in the waiting queue. After the interval, IGG_i orders the contracts in ascending order of price and for each contract IGG_i evaluates whether there are requests that can be redirected to the peer IGG. Fig. 3 illustrates the negotiation between IGG_i and IGG_j under a price-range contract. The scenario is as follows:

(1) IGG_i sends an offer to IGG_j when IGG_i's unit cost for the request is higher than the minimum price of the contract with IGG_j. The price in the offer p_{offer} is the minimum price specified in the contract between IGG_i and IGG_j.
(2) IGG_j, in turn, replies with one of the following messages:
  (2.1) IGG_j sends an accept message whose price is the price in the initial offer if the request's cost is lower than or equals to the amount that IGG_i is willing to pay (i.e. p_{offer} multiplied by the number of resource units required by the request rq_{units}).
  (2.2) If IGG_j's request cost is greater than the amount offered by IGG_i, but less than the maximum amount that IGG_i would possibly pay (i.e. the contract's maximum price p_{max} multiplied by rq_{units}), then it sends a counter-offer whose price is mc_j/rq_{units}. For simplicity the counter-offer contains the peering IGG_j's unit cost for the request, but the mechanism can easily be extended to incorporate a profit margin or use profit maximisation techniques.
  (2.3) If IGG_j's request cost is higher than the maximum amount IGG_i is willing to pay, the offer is rejected.
(3) After receiving IGG_j's message, IGG_i replies as follows:
  (3.1) IGG_i accepts the counter-offer if its request cost is still higher than the amount asked by IGG_j (i.e. number of resource units required rq_{units} multiplied by the counter-offer's price p_{ct}.
  (3.2) Otherwise, the counter-offer is rejected. IGG_i keeps the request in the queue and repeats the whole process for the next contract.

IGG_j stores the offers and evaluates them at time intervals. The evaluation algorithm sorts the offers by decreasing order of price. In addition, IGG_j maintains a list of tickets which it has created to

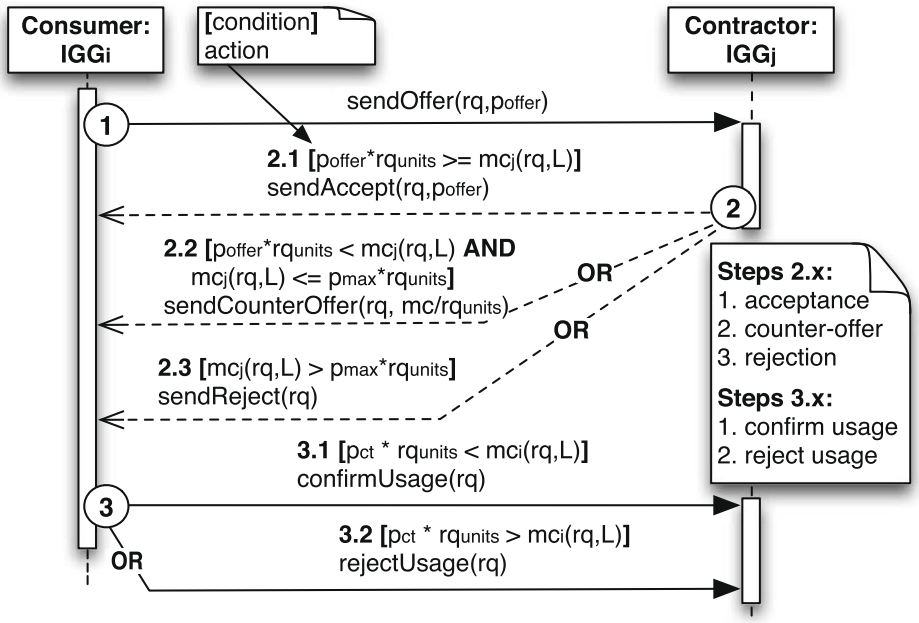

Fig. 3. Redirection negotiation.

serve the requests whose negotiation is in progress. This way, the evaluation of the request cost considers the requests being served as well as those whose negotiation is in progress. Creating a ticket corresponds to finding a time slot for the job. Moreover, in order to reduce the number of messages exchanged by IGGs, when IGG_i sends an offer to IGG_j, the offer contains a list of requests that IGG_i is willing to redirect to IGG_j. That is, a negotiation is performed for a group of requests and not on a per-request basis. IGG_j can accept all or part of the requests whose price is within the accepted price-range.

As described beforehand, there are two types of requests, namely best effort and deadline constrained. We use an earliest start time policy to select the resources to serve a request. The request's deadline is the time horizon used to calculate the load in the Grid, the load imposed by the request and consequently the request cost. This way, the Grid load for example, is determined by the resource shares provided by RPs and the allocations until the horizon. For best effort requests we create a virtual deadline given by the latest start time based on the time slots held by the gateway plus the runtime estimate; the virtual deadline is used as the horizon.

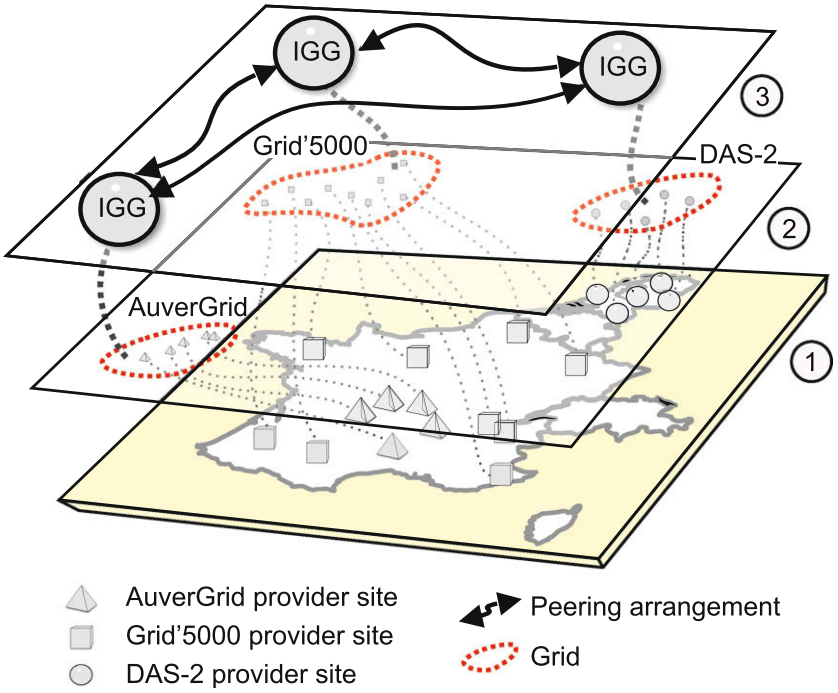

Fig. 4. Contract topology simulated.

5.3. Storing free time slots at the IGG

The resource providers issue free time slots and send them to the IGG on a periodical basis. The IGG maintains the availability information given by a provider on a modified red-black tree [55]. Each node has two references namely to its predecessor and successor nodes thus forming a linked list. This tree is analogous to the availability profile described by Mu'alem and Feitelson [19]; the nodes are ordered according to their times. That is, a free slot may lead to the creation of two nodes in the tree, namely to mark its start and finish times; free time slots can share nodes. We use the red-black tree to find a time slot to serve a job and the list to iterate the nodes and check whether the selected processing elements are available until the job's completion time.

6. Performance evaluation

6.1. Experimental scenario

The simulated environment is composed of three Grids, namely DAS-2 in the Netherlands and Grid'5000 and AuverGrid in France. The Grids DAS-2 [30], Grid'5000 [29] and AuverGrid [31] comprise 5, 15 and 5 clusters, respectively. For detailed information on the characteristics of the clusters we refer to Iosup et al. [8] and the Grid Workloads Archive website.[3] Fig. 4 presents the environment simulated.

The evaluation is performed through simulation by using a modified version of GridSim.[4] We resort to simulation because it provides a controllable environment and enables us to carry out repeatable experiments.

The workloads of the Grids are modelled using traces obtained from the Grid Workloads Archive. We divide the traces into 2-month intervals. At each simulation run, we randomly select one interval from the trace of each Grid to model the load of that Grid. Moreover, we disregard the first 8 months of DAS-2's trace, the first 4 months of AuverGrid's and the first 16 months of Grid'5000's. These intervals have not been considered because they may represent the set up phase of the Grids and the jobs in these intervals may not be representative of the Grids' workloads. We attempt to eliminate the system warm-up by disregarding the first two weeks of the experiments. For the load forecast policy, the second week is used for training.

The resource providers' local jobs are generated according to the workload model proposed by Lublin and Feitelson [56]; we refer to this model as Lublin99. We configure the Lublin99 model to generate type-less jobs (i.e. we do not make distinctions between batch and interactive jobs); the maximum number of CPUs used by the generated jobs is set accordingly to the number of nodes in the clusters; we generate 2-month long workloads. We change two parameters of the Lublin99 model when generating the workload for each cluster. The medium size of a parallel job (specified in log_2) is set to $log_2 m - \theta$ where m is the number of CPUs in the system and θ is drawn uniformly from 1.5 to 3.5. In addition, the inter-arrival rate of jobs is modified by setting the β of the used gamma distribution to a value uniformly distributed between 0.4871 and 0.55. These changes lead to workloads with different loads and different arrival rates, which we believe is representative of Grid resources. For load forecasting we use a weighted exponential moving average [57], considering a window of 25 intervals.

We perform experiments considering L in Eq. 4 equals to 95% of utilisation and k set to 5% of the Grid's resources. In this case, when the fixed-price ($price_{max}$) of a contract is the marginal cost of accepting a request requiring one resource unit of the Grid's capacity when the Grid is 95% utilised. The price-range contract has a maximum price of $price_{max}$ and a minimum price given by $price_{max}$ minus the difference between the request marginal cost at 95% and at 85% of utilisation.

6.2. Performance metrics

The performance evaluation considers two metrics: the average weighted response time (AWRT) [58] of jobs and the percentage of the generated load redirected by the IGGs. The AWRT measures how long in average users wait to have their jobs executed. A short AWRT indicates that on average users do not wait long for their jobs to complete. The redirected load demonstrates the performance of the mechanism in terms of managing peak loads; the AWRT, on the other hand, demonstrates whether the response time of user requests is improved through peering of IGGs or not.

$$AWRT_k = \frac{\sum_{j \in \tau_k} p_j \cdot m_j \cdot (c_j - s_j)}{\sum_{j \in \tau_k} p_j \cdot m_j} \quad (5)$$

The $AWRT_k$ relative to all jobs $j \in \tau_k$ that have been initially submitted to entity k is given by Eq. 5, where m_j is the number of processors required by job j, p_j is the execution time of the job, c_j is the time of completion of the job and s_j is its submission time. The resource consumption ($p_j \cdot m_j$) of each job j is used as the weight.

6.3. Policy acronyms

In order to reduce space, we abbreviate the name of the policies as follows. A policy name comprises two parts separated by +. The first part represents the policy employed by the provider whereas the second represents the gateway policy. In the resource provider's side, **Eb** stands for EASY backfilling, **Cb** for Conservative backfilling, **M** for Multiple partitions and **Mf** for Multiple partitions with load forecast. On the other side, for the gateway's policy, **least-load** means 'submit to least loaded resource', **earliest** represents 'select the earliest start time' based on the free time slots given by providers on a periodical basis. This way, **EbMf + earliest – partial** for example, indicates that providers use EASY backfilling, multiple partitions and load forecasts, whereas the gateway submits jobs selecting the earliest start time based on the availability information sent by providers at regular intervals.

6.4. Experimental results

The parameters used for the experiments are summarized in Table 1. The fixed cost of a resource in Eq. 1 is drawn uniformly

Table 1
Description of the parameters used in the experiments.

Parameter	Description
Number of Grids	3
Contract topology	all-to-all (see Fig. 4)
Number of simulation rounds	10
Cost of a resource unit	0.90–1.00
Load threshold (%)	95
Value of k (%)	5
Time between contract updates (hours)	1–6
Number of clusters at DAS-2	5
Number of CPUs at DAS-2	400
Number of clusters at AuverGrid	5
Number of CPUs at AuverGrid	475
Number of clusters at Grid'5000	15
Number of CPUs at Grid'5000	1368

[3] More details about the modelled resources and the traces used can be obtained from the Grid Workloads Archive at http://gwa.ewi.tudelft.nl/pmwiki/.

[4] More information about the changes in the simulator is available at http://www.gridbus.org/intergrid/gridsim.html.

from 0.9 to 1. The load threshold (L) and k are set to 95% and 5%, respectively. The IGGs inform one another about the fixed-prices or the price-ranges in their contracts based on the current resource demand at intervals between 1 and 6 h. The results are averages of 10 simulation rounds excluding the best and worst results. The simulation seed to generate the providers' local workloads, the prices and the contract update intervals is changed at each round.

6.4.1. First experiment

The first experiment evaluates the AWRT of both Grid and local jobs in a scenario wherein the providers send the availability information to the IGG every 12 h. Fig. 5 shows the AWRT of Grid applications for four sets of allocation policies (i.e. Eb + least – load and EbMf+, Cb+ and CbM + earliest – start). The initial four bars represent the AWRT under no peering between IGGs, that is, the IGGs have no contracts with one another and therefore do not redirect requests. Bars 5–7 represent the AWRT of Grid jobs when fixed-price contracts are established amongst IGGs, whereas bars 8–10 show the AWRT under price-range contracts. The EASY backfilling with 'submit to the least loaded resource' (i.e. bar 1) is shown for the sake of comparison. We observe that in an overall, the AWRT is reduced by the peering of Grids under both fixed-price and price-range contracts. This occurs despite the fact that IGGs accumulate a number of requests to be handled at random intervals between 1 and 5 min when contracts exist, in contrast to Eb + least – load in which requests are handled upon their arrival at the gateway. The load forecast based policy (EbMf + earliest – start) leads to a decrease in the AWRT of Grid jobs in both fixed-price and price-range contracts, but it does not perform as good as the conservative backfilling based policies. However, our initial expectations were that this policy would have less impact on the providers' local jobs because they resize the free time slots

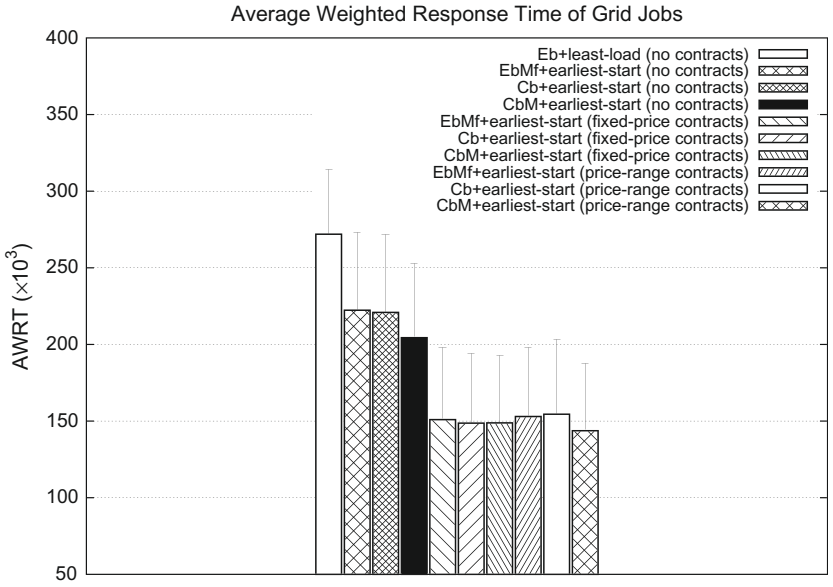

Fig. 5. Average weighted response time (AWRT) of Grid jobs.

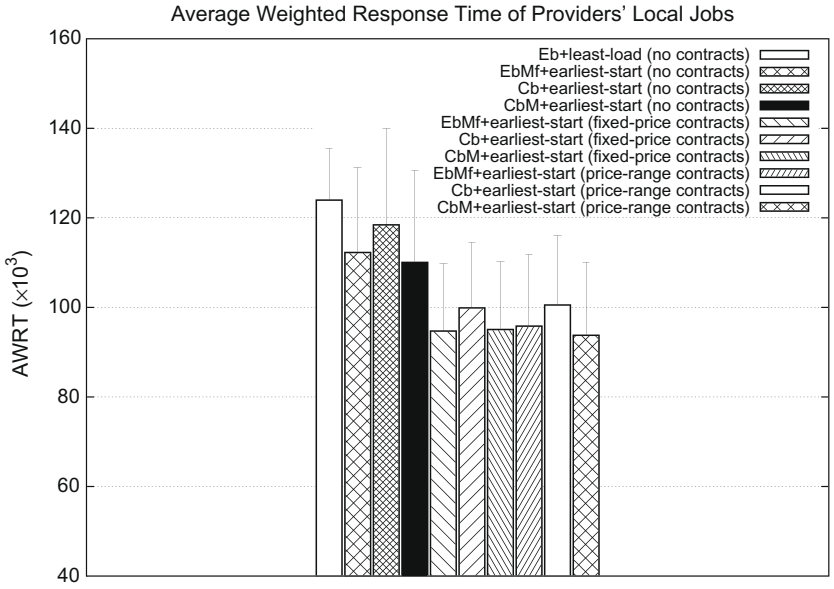

Fig. 6. Average weighted response time (AWRT) of providers' jobs.

given to the gateway based on load forecasts. In addition, previous results have shown that the load forecast policy is influenced by the length of the horizon [18].

The AWRT of local jobs show the impact of peering of Grids in the providers' user applications (Fig. 6). Similarly to the Grid applications, the AWRT of local jobs is reduced with the peering of IGGs. The reduction is more accentuated for the load forecast based policy, confirming our expectations that by providing load forecasts, even if not very precise, the gateway can schedule jobs accounting for the providers' local load. Intriguingly, the AWRT of both Grid and local jobs under price-range contracts is similar to, and in same cases worse than, that of fixed-price contracts. Our initial expectation was that, although Grids can redirect more requests under price-range contracts, the increase in AWRT could be caused by the fact that IGGs handle the requests and offers at random intervals between 1 and 5 min. However, as described later, with the chosen price-range contract, some IGGs in fact redirect less requests, thus the increase in AWRT is caused by the fact that a Grid ends up handling requests locally after a period of unsuccessful negotiations. This scenario can be improved by introducing a buy-it-now mechanism where a Grid could make an offer for immediate access to resources [45]. However, the investigation of such a mechanism is not in the scope of this paper.

Fig. 7 presents the percentage of the load from each Grid migrated to other Grids when providers send availability information every 12 h. A previous investigation [11] revealed that the job acceptance is higher when the contracts define a price-range, which allows Grids to redirect more load. However, with a price-range defined by $k = 5\%$, Grids do not redirect more load in all the cases. For example, Fig. 7 shows that when providers use conservative backfilling without multiple partitions, DAS-2 and AuverGrid in fact redirect less load. We do not investigate the impact of different price ranges on provisioning under the all the policies described in this paper [11].

6.4.2. Second experiment

In the second experiment we evaluate the AWRT of Grid jobs in three situations wherein the providers send the availability information to the gateway firstly every 24 h, secondly every 12 and finally every 6 h. Table 2 shows the AWRT of Grid jobs per Grid under each scenario. In our previous study [11], we noticed that AuverGrid has a higher load than DAS-2 and Grid'5000. The table shows that Grids with a low utilisation (i.e. DAS-2 and Grid'5000) do not have a decrease in the AWRT of their Grid users' applications. In fact, we can notice that, the AWRT is worsened. In contrast, AuverGrid has a substantial reduction in the AWRT of its

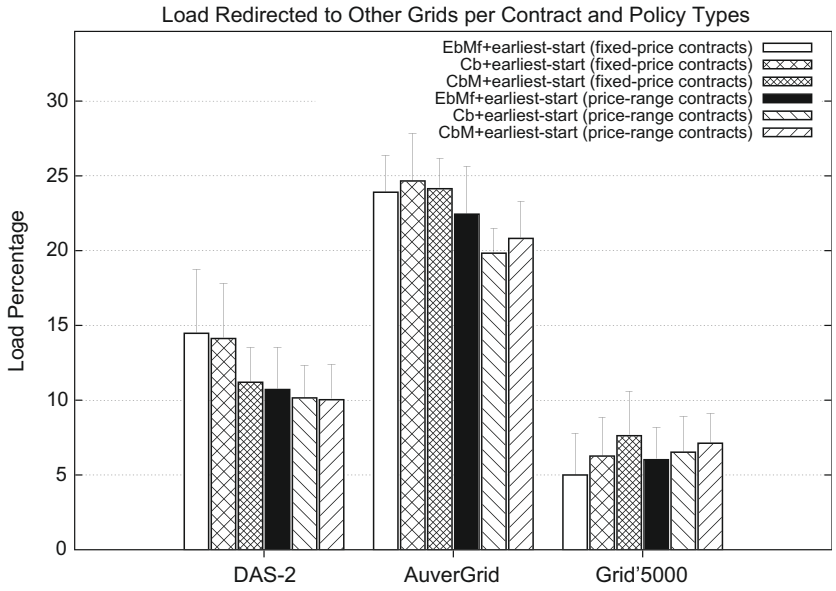

Fig. 7. Percentage of load generated by each Grid that was redirected to other Grids.

Table 2
The AWRT of Grid jobs under different policies and intervals.

Grid	No contracts			Fixed-price contracts			Price-range contracts		
	EbMf	Cb	CbM	EbMf	Cb	CbM	EbMf	Cb	CbM
	Providers sending availability information every 24 h								
DAS-2	39170	45644	43972	46025	50263	51258	47082	55502	47715
AuverGrid	419362	436861	394436	210710	200719	195419	193491	197075	207255
Grid5000	191685	181689	176156	194614	182955	177300	189055	181915	175958
	Providers sending availability information every 12 h								
DAS-2	37902	41295	40611	49190	44040	45101	50509	44800	43770
AuverGrid	433547	432191	391018	208548	218451	219517	214272	226716	220365
Grid5000	190568	179463	174029	188503	178315	174342	188572	178078	173047
	Providers sending availability information every 6 h								
DAS-2	39344	39063	38788	53297	42468	41154	51907	42818	40771
AuverGrid	439807	427912	389705	233263	240290	216316	227300	225625	215831
Grid5000	189918	174233	170628	189295	175202	168825	191194	173863	171537

Table 3
The AWRT of providers' local jobs under different policies and intervals.

Grid	No contracts			Fixed-price contracts			Price-range contracts		
	EbMf	Cb	CbM	EbMf	Cb	CbM	EbMf	Cb	CbM
	Providers sending availability information every 24 h								
DAS-2	62030	65112	61095	83605	87580	87795	80370	90657	78051
AuverGrid	287352	289922	262810	168008	163892	155444	152469	158253	160937
Grid5000	77573	76926	73146	76620	78199	72868	74580	78154	73939
	Providers sending availability information every 12 h								
DAS-2	60814	63256	60500	76554	82402	75124	81889	80374	73920
AuverGrid	269261	287900	261086	156361	172751	169217	166731	179381	169700
Grid5000	74285	75484	71831	73788	76657	71632	73263	75857	71478
	Providers sending availability information every 6 h								
DAS-2	61371	62180	59326	78647	77866	74680	79951	78018	72136
AuverGrid	284865	285926	260735	182447	190212	166531	170587	180327	169151
Grid5000	71993	74611	71098	72005	74084	70519	71216	74336	70774

Table 4
The AWRT of Grid jobs for an interconnection between DAS-2 and Grid'5000.

Grid	No contracts			Fixed-price contracts			Price-range contracts		
	EbMf	Cb	CbM	EbMf	Cb	CbM	EbMf	Cb	CbM
	Providers sending availability information every 24 h								
DAS-2	39139	45694	44001	40799	46058	45303	40623	46455	44495
Grid5000	140843	145003	142751	139269	146913	142429	139719	144835	142768
	Providers sending availability information every 12 h								
DAS-2	37881	41315	40560	39230	41330	41424	38984	41680	41806
Grid5000	138999	145607	140184	143663	143766	140379	141989	142437	140435
	Providers sending availability information every 6 h								
DAS-2	39325	39054	38741	39103	39427	38989	39410	39413	39043
Grid 5000	152476	141030	137761	139586	140367	137390	144104	140241	137596

Grid jobs. We can conclude that in terms of improving the AWRT, the peering of Grids with very different utilisation levels may not benefit the under-utilised Grids. However, the mechanism achieves its goal of redirecting requests from a Grid with high utilisation to others with lower utilisation levels as shown in Fig. 7.

During the second experiment, we also evaluated the AWRT of providers' local jobs at different Grids under different horizons. The results are presented in Table 3 and follow those of the AWRT of Grid jobs. AuverGrid benefits from the peering thus decreasing the AWRT of its providers' local jobs. Grid'5000 has small benefits in fixed price contracts when providers utilise a conservative backfilling policy with multiple partitions. DAS-2, on the hand, has the AWRT of its providers' job worsened by the peering.

6.4.3. Third experiment

The third experiment has the same characteristics of the second, except that we now investigate the peering only between DAS-2 and Grid'5000. With this experiment we want to investigate the AWRT of Grid jobs in the peering of two Grids that are not as utilised as AuverGrid. Table 4 shows the results for the AWRT of Grid jobs. The AWRT of Grid'5000's jobs is improved in some cases, for example, when the horizon is of 6 h. DAS-2 has small increases in the AWRT under the same horizon and policies. For the other horizons (i.e. 12 h and 24 h), the results are slightly mixed, presenting small improvements and some increases. The small increases are due to the fact that a gateway stores the messages to be handled at time intervals when the they have contracts with other gateways and some requests have an additional time incurred by the negotiation.

The experiments show that load management across Grids through resource exchange considering the compensation of resource providers is possible. The amount of load migrated shows that Grids balance their load and redirect requests. The allocation policies allow gateways to make decisions on resources provided to peering Grids. In addition, the overall AWRT of both Grid jobs and providers' local jobs is improved. However, some Grids have increases in the AWRT incurred by the negotiation time.

7. Conclusions and future work

This paper has presented the performance evaluation of policies for resource provisioning across Grids. It has demonstrated how a Grid can redirect requests to other Grids during periods of peak demand using a cost-aware load sharing mechanism. The mechanism relies on availability information obtained via different scheduling policies at provider sites. The provider policies enable information about fragments in the scheduling queue of clusters to be obtained using ordinary resource management systems. We have utilised this information as the basis for the mechanism for load sharing among Grids.

We have presented simulation results that demonstrate that the mechanism and policies are effective to redirect requests across Grids leading to a reduction in the overall average weighted response time (AWRT) of Grid applications. Moreover, we evaluate whether the proposed policies have a smaller impact on local providers' jobs than traditional policies when the Grids are interconnected. We have noticed that in an overall, the AWRT of providers' local jobs improves with a network of contracts among the Grids. However, some Grids have increases in the AWRT incurred by the negotiation time. The experiments demonstrate that, despite the imprecise resource availability information given by providers, the load management across Grids through resource exchange is possible while accounting for the compensation of resource providers.

Future work will investigate a buy-it-now mechanism for requests that have deadlines in order to reduce the negotiation time

required to acquire resources to serve them. We will also consider requests with different priorities. We also plan to investigate how IGGs can co-ordinate resource provisioning via shared spaces implemented atop distributed hash tables (DHT) or other P2P systems. We will extend the mechanism by providing means for Grids to redirect requests across several Grids (i.e. it will support transitive relationships between the Grids in the contract network). Future investigations also include more sophisticated resource provisioning policies for the gateways, specially for handling advance reservation requests, more sophisticated load forecasting techniques and the impact of varying price-range contracts on provisioning.

Acknowledgement

We would like to thank: the anonymous reviewers for their comments; Marco Netto, Sungjin Choi and Alexandre di Costanzo from the University of Melbourne for the technical discussions on the topic; Mukaddim Pathan for helping in improving the language and expression of a preliminary version of this paper. We are also grateful to Dr. Franck Cappello, Dr. Olivier Richard, Dr. Emmanuel Medernach and the Grid Workloads Archive group for making the Grid workload traces available. This work is supported by DEST and ARC Project grants. Marcos' PhD research is partially supported by National ICT Australia (NICTA).

References

[1] C. Catlett, P. Beckman, D. Skow, I. Foster, Creating and operating national-scale cyberinfrastructure services, Cyberinfrastructure Technology Watch Quarterly 2 (2) (2006) 2–10.
[2] T. Dunning, R. Nandkumar, International cyberinfrastructure: Activities around the globe, Cyberinfrastructure Technology Watch Quarterly, 2(1), URL <http://www.ctwatch.org/quarterly/articles/2006/02>.
[3] K. Miura, Overview of Japanese science Grid project NAREGI, Progress in Informatics (2006) 67–75.
[4] Open Science Grid, <http://www.opensciencegrid.org>, 2005.
[5] L. Peterson, S. Muir, T. Roscoe, A. Klingaman, PlanetLab Architecture: An Overview, Technical Report PDN-06-031, PlanetLab Consortium, Princeton, USA, May 2006.
[6] I. Foster, C. Kesselman, S. Tuecke, The anatomy of the grid: enabling scalable virtual organizations, International Journal of Supercomputer Applications 15 (3) (2001) 200–222.
[7] Grid Interoperability now community group (GIN-CG), <http://forge.ogf.org/sf/projects/gin>, 2006, URL http://forge.ogf.org/sf/projects/gin.
[8] A. Iosup, D.H.J. Epema, T. Tannenbaum, M. Farrellee, M. Livny, Inter-operating grids through delegated matchmaking, in: 2007 ACM/IEEE Conference on Supercomputing (SC 2007), Reno, USA, 2007.
[9] C. Metz, Interconnecting ISP networks, IEEE Internet Computing 5 (2) (2001) 74–80.
[10] P. Baake, T. Wichmann, On the economics of internet peering, NETNOMICS 1 (1) (1999) 89–105.
[11] M.D. de Assunção, R. Buyya, A cost-aware resource exchange mechanism for load management across Grids, in: 14th IEEE International Conference on Parallel and Distributed Systems (ICPADS'08), Melbourne, Australia, 2008.
[12] OpenPBS: The portable batch system software, Veridian Systems, Inc., Mountain View, CA, 2005, URL <http://www.openpbs.org/scheduler.html>.
[13] The Condor Project homepage, <http://www.cs.wisc.edu/condor/>, 2005.
[14] D.A. Lifka, The scheduling systems, in: Workshop on Job Scheduling Strategies for Parallel Processing (IPPS'95), Springer-Verlag, London, UK, 1995, pp. 295–303.
[15] R. Buyya, D. Abramson, J. Giddy, Nimrod/g: An architecture for a resource management and scheduling system in a global computational Grids, in: Fourth International Conference on High Performance Computing in Asia–Pacific Region (HPC Asia 2000), Beijing, China, 2000, pp. 283–289.
[16] S. Venugopal, R. Buyya, L. Winton, A grid service broker for scheduling e-science applications on global data grids: research articles, Concurrency and Computation: Practice and Experience (CCPE) 18 (6) (2006) 685–699.
[17] G. Singh, C. Kesselman, E. Deelman, A provisioning model and its comparison with best-effort for performance-cost optimization in Grids, in: 16th International Symposium on High Performance Distributed Computing (HPDC 2007), ACM Press, Monterey, USA, 2007, pp. 117–126.
[18] M.D. de Assunção, R. Buyya, Performance analysis of multiple site resource provisioning: Effects of the precision of availability information, in: International Conference on High Performance Computing (HiPC 2008), Bangalore, India, 2008.
[19] A.W. Mu'alem, D.G. Feitelson, Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling, IEEE Transactions on Parallel and Distributed Systems 12 (6) (2001) 529–543.
[20] B.G. Lawson, E. Smirni, Multiple-queue backfilling scheduling with priorities and reservations for parallel systems, in: Eighth International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP'02), LNCS, Springer-Verlag, London, UK, 2002, pp. 72–87.
[21] A. AuYoung, L. Grit, J. Wiener, J. Wilkes, Service contracts and aggregate utility functions, in: 15th IEEE International Symposium on High Performance Distributed Computing (HPDC 2006), Paris, France, 2006, pp. 119–131.
[22] T. Röblitz, F. Schintke, J. Wendler, Elastic Grid reservations with user-defined optimization policies, in: Workshop on Adaptive Grid Middleware (AGridM 2004), Antibes Juan-les-Pins, France, 2004.
[23] M. Wieczorek, M. Siddiqui, A. Villazon, R. Prodan, T. Fahringer, Applying advance reservation to increase predictability of workflow execution on the Grid, in: Second IEEE International Conference on e-Science and Grid Computing (E-Science 2006), IEEE Computer Society, Washington, DC, USA, 2006, p. 82.
[24] P. Padala, K.G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, K. Salem, Adaptive control of virtualized resources in utility computing environments, in: The 2007 Conference on EuroSys (EuroSys 2007), ACM Press, Lisbon, Portugal, 2007, pp. 289–302.
[25] P. Garbacki, V.K. Naik, Efficient resource virtualization and sharing strategies for heterogeneous Grid environments, in: Tenth IFIP/IEEE International Symposium on Integrated Network Management (IM 2007), Munich, Germany, 2007, pp. 40–49.
[26] G. Singh, C. Kesselman, E. Deelman, Application-level resource provisioning on the grid, in: Second IEEE International Conference on e-Science and Grid Computing (e-Science 2006), Amsterdam, The Netherlands, 2006, pp. 83–83.
[27] D.B. Jackson, Q. Snell, M.J. Clement, Core algorithms of the Maui scheduler, in: 7th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP'01), LNCS, Springer-Verlag, London, UK, 2001, pp. 87–102.
[28] CNGrid project web site, <http://www.cngrid.org/>, 2007.
[29] R. Bolze, F. Cappello, E. Caron, M. Daydé, F. Desprez, E. Jeannot, Y. Jégou, S. Lantéri, J. Leduc, N. Melab, G. Mornet, R. Namyst, P. Primet, B. Quetier, O. Richard, E.-G. Talbi, T. Iréa, Grid'5000: a large scale and highly reconfigurable experimental Grid testbed, International Journal of High Performance Computing Applications 20 (4) (2006) 481–494.
[30] The Distributed ASCI Supercomputer 2 (DAS-2), Dutch University Backbone, 2006.
[31] Conseil Régional Auvergne, AuverGrid, <http://www.auvergrid.fr>, 2007.
[32] Onelab2 website, <http://www.one-lab-2.org/>, 2007.
[33] L. Peterson, J. Wroclawski, Overview of the GENI architecture, GENI Design Document GDD-06-11, GENI: Global Environment for Network Innovations, January 2007, URL <http://www.geni.net/GDD/GDD-06-11.pdf>.
[34] Y. Wang, D. Scardaci, B. Yan, Y. Huang, Interconnect EGEE and CNGRID e-infrastructures through interoperability between gLite and GOS middlewares, in: International Grid Interoperability and Interoperation Workshop (IGIIW 2007) with e-Science 2007, IEEE Computer Society, Bangalore, India, 2007, pp. 553–560.
[35] D. Irwin, J. Chase, L. Grit, A. Yumerefendi, D. Becker, K.G. Yocum, Sharing networked resources with brokered leases, in: USENIX Annual Technical Conference, Boston, MA, 2006, pp. 199–212.
[36] L. Ramakrishnan, D. Irwin, L. Grit, A. Yumerefendi, A. Iamnitchi, J. Chase, Toward a doctrine of containment: Grid hosting with adaptive resource control, in: 2006 ACM/IEEE Conference on Supercomputing (SC 2006), ACM Press, New York, NY, USA, 2006, p. 101.
[37] L. Grit, D. Inwin, A. Yumerefendi, J. Chase, Virtual machine hosting for networked clusters: building the foundations for 'autonomic' orchestration, in: First International Workshop on Virtualization Technology in Distributed Computing (VTDC 2006), Tampa, Florida, 2006.
[38] R. Ranjan, A. Harwood, R. Buyya, SLA-based coordinated superscheduling scheme for computational Grids, in: IEEE International Conference on Cluster Computing (Cluster 2006), Barcelona, Spain, 2006, pp. 1–8.
[39] M. Balazinska, H. Balakrishnan, M. Stonebraker, Contract-based load management in federated distributed systems, in: First Symposium on Networked Systems Design and Implementation (NSDI), USENIX, San Francisco, CA, 2004, pp. 197–210.
[40] Y.-T. Wang, R.J.T. Morris, Load sharing in distributed systems, IEEE Transactions on Computers C-34 (3) (1985) 204–217.
[41] S. Surana, B. Godfrey, K. Lakshminarayanan, R. Karp, I. Stoica, Load balancing in dynamic structured peer-to-peer systems, Performance Evaluation 63 (3) (2006) 217–240.
[42] K. Lai, L. Rasmusson, E. Adar, S. Sorkin, L. Zhang, B.A. Huberman, Tycoon: an implementation of a distributed market-based resource allocation systems, Technical Report, HP Labs, Palo Alto, CA, USA, December 2004, URL <http://www.hpl.hp.com/research/tycoon/doc/csDC0412038.pdf>.
[43] T. Eymann, O. Ardaiz, M. Catalano, P. Chacin, I. Chao, F. Freitag, M. Gallegati, G. Giulioni, L. Joita, L. Navarro, D.G. Neumann, O. Rana, M. Reinicke, R.C. Schiaffino, B. Schnizler, W. Streitberger, D. Veit, F. Zini, Catallaxy-based grid markets, International Journal on Multiagent and Grid Systems, Special Issue on Smart Grid Technologies and Market Models 1 (4) (2005) 297–307.

[44] J. Brunelle, P. Hurst, J. Huth, L. Kang, C. Ng, D. Parkes, M. Seltzer, J. Shank, S. Youssef, Egg: An extensible and economics-inspired open grid computing platform, in: Third International Workshop on Grid Economics and Business Models (GECON 2006), Singapore, 2006, pp. 140–150.
[45] A. AuYoung, B. Chun, C. Ng, D.C. Parkes, A. Vahdat, A. Snoeren, Practical market-based resource allocation, Technical Report CS2007-0901, CSE, University of California San Diego, 2007.
[46] C. Ernemann, V. Hamscher, R. Yahyapour, Economic scheduling in grid computing, in: Revised Papers from the Eighth International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2002), Springer-Verlag, London, UK, 2002, pp. 128–152.
[47] R. Buyya, Economic-based distributed resource management and scheduling for Grid computing, PhD Thesis, Monash University, Melbourne, Australia, April 2002, URL <http://www.buyya.com/thesis/>.
[48] R.K. Dash, N.R. Jennings, D.C. Parkes, Computational-mechanism design: a call to arms, IEEE Intelligent Systems 18 (6) (2003) 40–47.
[49] J.S. Rosenschein, G. Zlotkin, Rules of Encounter: Designing Conventions for Automated Negotiation among Computers, The MIT Press, Cambridge, 1994.
[50] M.D. de Assunção, R. Buyya, S. Venugopal, InterGrid: A case for internetworking islands of Grids, Concurrency and Computation: Practice and Experience (CCPE) 20 (8) (2008) 997–1024.
[51] Y. Fu, J. Chase, B. Chun, S. Schwab, A. Vahdat, SHARP: An architecture for secure resource peering, in: 19th ACM Symposium on Operating Systems Principles (SOSP 2003), New York, NY, USA, 2003, pp. 133–148.
[52] D.D. Clark, J. Wroclawski, K.R. Sollins, R. Braden, Tussle in cyberspace: defining tomorrow's internet, IEEE/ACM Transactions on Networking 13 (3) (2005) 462–475.
[53] M.B. Weiss, S.J. Shin, Internet interconnection economic model and its analysis: peering and settlement, NETNOMICS 6 (1) (2004) 43–57.
[54] N. Badasyan, S. Chakrabarti, Private peering, transit and traffic diversion, NETNOMICS 7 (2) (2005) 115–124.
[55] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, second ed., MIT Press/McGraw-Hill, Cambridge, Massachusetts, 2001.
[56] U. Lublin, D.G. Feitelson, The workload on parallel supercomputers: modeling the characteristics of rigid jobs, Journal of Parallel and Distributed Computing 63 (11) (2003) 1105–1122.
[57] J.E. Hanke, A.G. Reitsch, Business Forecasting, fifth ed., Prentice-Hall Inc., Englewood Cliffs, USA, 1995.
[58] C. Grimme, J. Lepping, A. Papaspyrou, Prospects of collaboration between compute providers by means of job interchange, Job Scheduling Strategies for Parallel Processing, of Lecture Notes in Computer Science, 4942, Springer, Berlin/Heidelberg, 2008, pp. 132–151.