A Cost-Aware Resource Exchange Mechanism for Load Management across Grids

Marcos Dias de Assunção[1,2] and Rajkumar Buyya[1]
[1] Grid Computing and Distributed Systems (GRIDS) Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
{marcosd, raj}@csse.unimelb.edu.au

[2] NICTA Victoria Research Laboratory
The University of Melbourne, Australia

Abstract

Numerous Grids have been created during the last years. Most of these Grids work in isolation and with different utilisation levels. As the resource utilisation within a Grid has fixed and operational costs, there can be benefits for a Grid to offload requests to another Grid or provide spare resources, thus reducing the cost of over-provisioning. In this work, we enable load management across Grids through resource exchange between them considering the cost for one Grid to acquire resources from another. However, enabling resource exchange amongst Grids is a challenging task: a Grid should not compromise the performance of its local user communities' applications, yet benefit from providing spare resources to other Grids. The load management mechanism and related policies take into consideration the economic compensation of providers for the resources allocated. Experimental results show that the mechanism achieves its goal in redirecting requests, increasing the number of user requests accepted and balancing the load amongst Grids.

1. Introduction

The potential of Grid computing has led to the creation of several Grid-based resource sharing networks (e.g. TeraGrid [5], Naregi [19] and Open Science Grid [21]). Although these Grids have been used for various applications, they mostly work in isolation and with different utilisation levels [14], thus contrasting with the original view of Grid computing [10]. The Grid Interoperability Now - Community Group (GIN-CG) [12] has been working to provide interoperability between Grids by developing components and adapters to enable secure job submissions, data transfers and information queries. These efforts are crucial to enable interoperability between Grids, but GIN-CG also highlights the need for resource management across Grids.[1]

A Grid infrastructure is expensive to maintain as resource utilisation within it has fixed and operational costs such as those with electricity providers and system administrators. Resource providers may offer resources to the Grid expecting some economic compensation. Consequently, 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. This load management across Grids could reduce the costs incurred by over-provisioning.

Enabling load management across Grids, however, is complex due to the autonomy for capacity planning and provisioning of resources to user communities within each Grid. There is contention for resources and dynamicity of shares supplied by resource providers within each Grid. The main challenges when designing a load management mechanism are how a Grid can (i) meet the demand of local user communities by providing the required resources; (ii) coordinate with other Grids through peering arrangements or contracts to acquire additional resources to satisfy excess demands; and (iii) provide spare resources to other Grids in return for payments.

In this work, we focus on load management across Grids and investigate a resource exchange mechanism that takes into account the economic compensation of resource providers and considers the cost for one Grid to acquire computational resources from another. Grids establish contracts that resemble the peering arrangements between Internet Service Providers (ISPs). The Internet is composed of competing ISPs that agree to allow traffic into one another's networks providing their customers with connectivity to the

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

entire Internet. These agreements between ISPs are commonly termed as peering and transit arrangements [18]. Here, Grids establish arrangements specifying the type of resource exchanged as well as their price.

The proposed mechanism allows a peering Grid to redirect a request to another Grid when the cost of serving it locally is higher than the price the Grid would pay to the peering Grid to process the request. The redirection takes place between Grids that have a pre-established peering arrangement. We evaluate the proposed mechanism in terms of load balancing across the Grids and the increase in the overall request acceptance rate.

The main contributions of this paper are:

- A load management mechanism based on resource exchange and request redirection between Grids.
- Policies for redirection (acceptance) of requests across (from) peering Grids, which allow a Grid to evaluate when it is economic beneficial to redirect (accept) requests.

2. Related Work

Several Grid-based resource sharing networks have been built over past few years [3, 5–7, 19, 21, 27]. Recently, initiatives have emerged for linking resource sharing networks. OneLab2 [20] and the Global Environment for Network Innovations (GENI) [23] have evolved from the PlanetLab architecture [22] to allow the federation of autonomous networks controlled by different organisations. GIN-CG under the Open Grid Forum [12] utilises community efforts to address issues on security, standard job submission, data management and information services. Other Grid middleware interoperability approaches have also been presented [28]. These efforts provide the basis for load management across Grids by facilitating standard job submission and request redirection. We attempt to build on these to investigate resource management across Grids.

Shirako [15, 24] is a resource management system based on a resource leasing abstraction. This system allows sites to 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. Our work shares some concepts with Shirako such as the delegation of provisioning rights. We focus on the policies for resource exchange amongst InterGrid Gateways, which are similar to brokers in Shirako. To the best of our knowledge, resource exchange amongst Shirako brokers has not been explored yet. Grit *et al.* [13] investigate the number of Virtual Machine (VM) migrations required when a broker and provider sites use either conflicting or synchronised policies for resource provisioning. 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 the load of resource providers as an on-off model described later [1]. 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.

Iosup *et al.* [14] 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 economic compensation for the resources acquired from the providers.

A Service Level Agreement (SLA) based co-ordination mechanism for a federation of clusters of computers was proposed by Ranjan *et al.* [25]. 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.

The mechanism presented in this work derives from Medusa [2], 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.

Wang and Morris [29] demonstrated that efficiency in load sharing depends on the environment. In addition, server-initiated tends to out-perform source-initiated strategies when the same amount of information about stakeholders is available. Their experiments consider single processor nodes in a local network. In our scenario, resources have multiple processors; also, resources are heterogeneous in the number of processors. These make the local scheduling sub-problem different; in addition, resource management across Grids introduces a third subproblem: the load sharing between Grids. Surana *et al.* [26] 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. Load balancing is achieved by migrations of virtual servers. Although this technique has a very low overhead, we do not perform migration of virtual

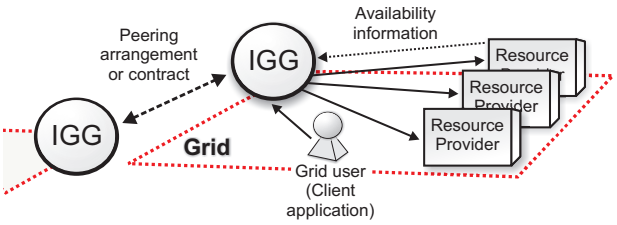

Figure 1. The main InterGrid components.

servers, and focus on the request redirection and their assignment to resources.

3. InterGrid: Resource Exchange across Grids

Our previous work 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 another work [9].

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 [18]. The resources exchanged between Grids can be physical or virtual resources such as Virtual Machines (VMs). Figure 1 provides an overview of the InterGrid architecture.

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 advertises the resource by delegating its provisioning rights over a given time frame. This delegation can use a secure protocol such as SHARP [11]. In this paper we consider one type of resource (i.e. CPU); the resource providers send the availability information in the form of free time slots. A free time slot contains the number of CPUs available, their configuration, and the start and finish times of the slot. For the sake of simplicity here, an RP delegates provisioning rights directly to an InterGrid Gateway (IGG). However, the architecture also considers the case wherein the Grid has already an internal broker or resource management system [9].

A Grid has peering arrangements with other Grids, managed by IGGs and, through which they coordinate the use of resources of the InterGrid.[2] 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 what peering Grid is selected to process a request and at what price the processing is performed.

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 peering 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 is contiguous and needs to be served with resources from a single resource provider. We do not handle co-allocations 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 economic compensation; and (iii) acquire resources from other Grids to satisfy its users under peak load conditions.

4. The Load Management Mechanism

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. The use of a quadratic function allows us to specify contracts with price ranges as discussed later in this section.

[2]The assumption of pre-established arrangements is reasonable as current Grids need to reserve the network links and set up the resources required. An example includes the interconnection of Grid'5000 with DAS-3 and NAREGI. More information is available at http://www.grid5000.fr

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 Equation 1.

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

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 Equation 2. If request u requires one resource unit, then the request cost is equals to a *unit cost*.

$$mc_i = cost_i(L + u_{load}) - cost_i(L) \quad (2)$$

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.

4.1. 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.* [2], 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 Equation 3.

$$\delta_k(L) = \frac{mc(u, L - u_{load}) - mc(u, L - k - u_{load})}{u_{units}} \quad (3)$$

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.

4.2. Request Redirection 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. Figure 2 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.

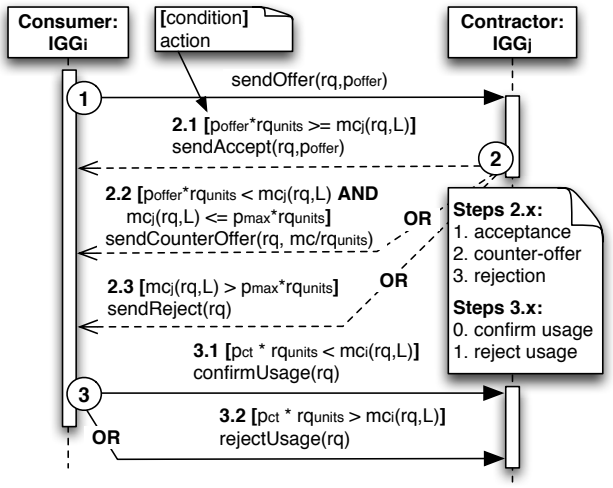

Figure 2. Redirection negotiation.

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 in the following manner:

   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 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 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.

5. Performance Evaluation

5.1. Experimental Scenario

The modelled environment is composed of three Grids, namely DAS-2 in the Netherlands and Grid'5000 and AuverGrid in France. The Grids DAS-2, Grid'5000 and AuverGrid comprise 5, 15 and 5 clusters respectively. For detailed information on the characteristics of the clusters we refer to the Grid Workloads Archive website.[3] Figure 3 presents the environment simulated (layers 1, 2 and 3 represent a pictorial view of the physical location of provider sites, their Grid organisation and the interGrid configuration respectively). Table 1 summarises the parameters.

Table 1. Parameter description.

Parameter	Value
Number of Grids	3
Contract topology	all-to-all (Figure 3)
Load at off-peak (ON) intervals (%)	50 – 100
Load at peak (OFF) intervals (%)	20 – 50
ON interval duration (hours)	24 – 72
OFF interval duration (hours)	12 – 48
Cost of a resource unit	0.90 – 1.00
Number of dedicated CPUs (DAS-2)	192
Number of dedicated CPUs (AuverGrid)	234
Number of dedicated CPUs (Grid'5000)	648
Deadline constrained requests (%)	30
Stringency factor	5.00

The evaluation of the proposed mechanism is performed

[3] http://gwa.ewi.tudelft.nl/pmwiki/

through simulation by using a modified version of GridSim [4].[4] We use simulation as it provides a controllable environment and enables us to carry out repeatable experiments. The resource providers use a pricing function for a resource unit given by $price = cost + (cost * load)$ where $cost$ is drawn from a uniform distribution; $load$ is generated using an on-off model as described by AuYoung *et al.* [1], wherein on and off intervals correspond to off-peak and peak periods respectively. The duration of these intervals and the load in each are also modelled using uniform distributions as described in Table 1. The on-off model is used to model the availability of part of the providers, around 50%, the remaining resource providers are dedicated to the gateways. Table 1 presents the number of CPUs dedicated in each Grid. We are currently investigating how to improve this scenario and obtain this provisioning information from traditional aggressive and conservative backfilling policies and through the use of multiple resource partitions [8, 17].

The workloads of the Grids use traces of real Grids obtained from the Grid Workloads Archive. We divided the traces into 4-month intervals. We use the interval between the 9th-12th months of DAS-2' trace, the 5th-8th months of AuverGrid's and the 17th-20th months of Grid'5000's. We make an effort to eliminate the cool-down phase of the system. This way, the last simulated event is the arrival of the last job submitted in any of the Grid workloads. Additionally, we attempt to eliminate the system warm-up by disregarding the first week of the experiments. As described beforehand, there are two types of requests, namely deadline constrained and best-effort. We select randomly the requests that are deadline constrained. In order to generate the request deadlines we use a technique described by Islam *et al.* [16], which provides a feasible schedule for the jobs. We perform the experiments by using the same Grid environment with no contracts amongst the gateways, using an aggressive backfilling policy at the resource providers and a 'submit to the least loaded resource' policy at the gateway. A request's deadline is the completion of the corresponding job under this scenario multiplied by a stringency factor (Table 1).

We calculate the price of a resource unit in the contracts between the Grids by assigning different values to L in Equation 3. We perform experiments considering L equals to 0.99 and 0.95 and with different values for k (i.e. 0.01, 0.05, 0.1 and 0.2). For example, when L=0.99 and k=0.01, the fixed price ($price_{max}$) of a contract is the cost of a request requiring one resource unit of the Grid's capacity when the Grid is 99% 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 cost at 99% and at 98% of utilisation.

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

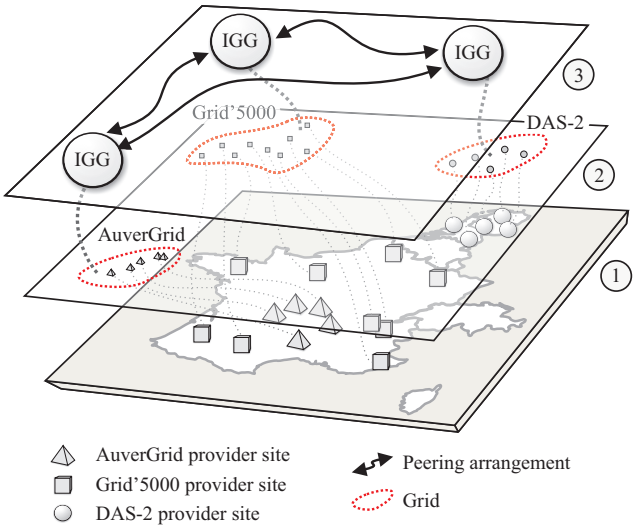

Figure 3. Contract topology simulated.

Performance metrics: We select two metrics, namely increase in requests accepted and percentage of the generated load redirected by the IGGs. The redirected load demonstrates the performance of the mechanism in terms of managing peak loads; the increase in requests accepted, on the other hand, demonstrates whether the IGG compromises local users by peering with other IGGs. We also compute the increase in utilisation for comparison against the migration of load.

5.2. Experimental Results

All the results presented in this section are averages of 10 simulation rounds using different seeds and excluding the best and worst results. The global increase in the number of requests served under different types of contracts is shown in Table 2. Overall, there is an increase in the number of requests accepted, except for DAS-2, whose acceptance rate is decreased. Further investigation revealed that DAS-2 has a lower utilisation than the other two Grids. When we calculate the deadlines using the 'submit to least loaded resource' and aggressive backfilling policies, the deadlines become very tight as many jobs are started immediately while others backfill easily thus generating very tight deadlines. As IGGs run the provisioning algorithm at time intervals, some requests are rejected. When the deadlines are not used, all the requests are processed. As the price range increases, more load can be migrated. However, in the experiments we performed, as k becomes large (e.g. $k > 0.2$), the mechanism becomes unstable as Grids tend to both redirect and accept too many requests.

The table also shows the increase in resource utilisation at each Grid. Also, Grid'5000 redirects smaller amounts of load (Figure 4). Even though it has not a substantial in-

Table 2. Increase in both requests accepted and resource utilisation.

Metric	Grid	$L = 0.99$					$L = 0.95$				
		Fixed Price	k				Fixed Price	k			
			0.01	0.05	0.1	0.2		0.01	0.05	0.1	0.2
Increase in number of requests served	DAS-2	-6.50	-4.50	-3.38	-5.25	-10.00	-5.38	-4.25	-8.00	-4.25	-4.88
	AuverGrid	658.00	661.12	661.12	657.00	663.00	650.38	659.50	658.12	661.00	662.00
	Grid´5000	19.12	8.88	10.62	4.88	6.38	18.25	7.00	13.12	10.50	14.88
Increase in resource utilisation (%)	DAS-2	7.62	7.99	9.42	12.56	8.39	7.72	8.93	8.18	8.69	9.24
	AuverGrid	-23.30	-24.15	-26.45	-27.56	-29.15	-22.20	-23.45	-25.42	-27.52	-30.96
	Grid´5000	6.12	6.38	6.92	6.35	7.98	5.71	5.51	6.78	7.25	8.13

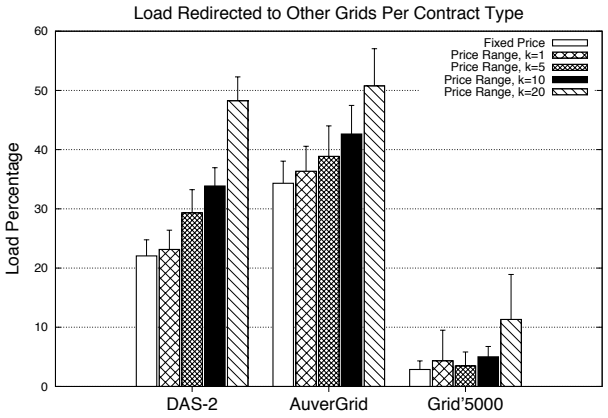

Figure 4. Load redirected (L=0.99).

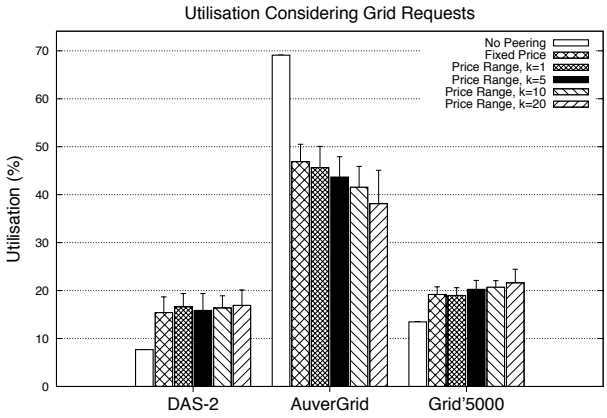

Figure 5. Resource utilisation at Grids (L=0.99).

crease in the acceptance rate of the requests originated by its users, it increases its resource utilisation without compromising the acceptance rate. This leads to an increase in profit as they can receive a payment from the other Grids for the resources provided. However, we do not measure the profits of each Grid here. Overall, the algorithms achieve their goal, which is to redirect requests driven by the requests cost. AuverGrid has a decrease in utilisation in contrast to DAS-2 and Grid'5000. As AuverGrid has a higher utilisation than DAS-2 and Grid'5000 when they are not redirecting requests, this decrease in utilisation shows that the mechanism is effective in redirecting AuverGrid's requests to other Grids. Figure 5 shows that there is an initial load imbalance between the Grids, as the utilisation of AuverGrid without contracts is close to 70%, while DAS-2 and Grid'5000 are both close to 10%. The table shows that the mechanism helps in balancing the load across the Grids connected.

Figure 4 presents the percentage from the load generated by each Grid migrated to other Grids, or exchanged by the IGGs. When Grids define the maximum price for a resource unit as the unit cost at 99% of utilisation (i.e. $L = 0.99$), they exchange load and the overall number of requests accepted is improved in almost all the cases (Table 2). The acceptance is better when the contracts define a price range, which allows Grids to redirect more load.

When IGGs set the maximum price as the unit cost at a low value for L and the price range is large, the number of requests accepted increases whereas the amount of load exchanged decreases. The reason for this behaviour is that overloaded IGGs tend to be more conservative in accepting load from other IGGs, even though they try to migrate load more easily. An IGG_i that needs to offload will redirect requests willing to pay a price lower or equal to the maximum price of a contract. If the unit cost of the IGG considering to accept the load is slightly above the maximum price, it will not accept the load. In our experiments, this behaviour is reflected in the increase of requests accepted.

The experiments show that load management across Grids through resource exchange considering the economic compensation of resource providers is possible. The resource utilisation and increase in requests accepted show that Grids balance their load and redirect requests thus minimising the costs with resource usage.

6. Conclusions and Future Work

This paper proposes a mechanism and policies to redirect requests across Grids during periods of peak demand. Simulation results demonstrate that the mechanism achieves its goal as it leads to an overall increase in requests accepted and balances the load across the interconnected Grids. The experiments show that load management across Grids through resource exchange taking into account the economic compensation of resource providers is possible.

In future work IGGs will be able to overbook resources and employ techniques for maximising their revenue. We also plan to include and handle penalties when providers cannot honour their resource offers. Moreover, the proposed mechanism will be extended by providing means for Grids to redirect requests across several Grids (i.e. we are extending the mechanism to support transitive relationships between the Grids in the contract network).

Acknowledgements

We thank Marco Netto, Kyong Hoon Kim, Suraj Pandey, Sungjin Choi, Mukaddim Pathan, Alexandre di Costanzo and Srikumar Venugopal from the University of Melbourne for sharing their thoughts on the topic. We are 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 NICTA.

References

[1] A. AuYoung, L. Grit, J. Wiener, and J. Wilkes. Service contracts and aggregate utility functions. In *15th IEEE International Symposium on High Performance Distributed Computing (HPDC 2006)*, pages 119–131, Paris, France, July 2006.

[2] M. Balazinska, H. Balakrishnan, and M. Stonebraker. Contract-based load management in federated distributed systems. In *1st Symposium on Networked Systems Design and Implementation (NSDI)*, pages 197–210, San Francisco, USA, March 2004.

[3] 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, and T. Iréa. Grid'5000: a large scale and highly reconfigurable experimental Grid testbed. *International Journal of High Performance Computing Applications*, 20(4):481–494, November 2006.

[4] R. Buyya and M. Murshed. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Practice and Experience (CCPE)*, 14(13–15):1175–1220, November-December 2002.

[5] C. Catlett, P. Beckman, D. Skow, and I. Foster. Creating and operating national-scale cyberinfrastructure services. *CTWatch Quarterly*, 2(2):2–10, May 2006.

[6] CNGrid. Project website. http://www.cngrid.org/, 2007.

[7] Conseil Régional Auvergne, AuverGrid. Project website. http://www.auvergrid.fr, 2007.

[8] M. D. de Assunção and 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, December 2008.

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

[10] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructures*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[11] Y. Fu, J. Chase, B. Chun, S. Schwab, and A. Vahdat. SHARP: An architecture for secure resource peering. In *19th ACM Symposium on Operating Systems Principles (SOSP 2003)*, pages 133–148, New York, NY, USA, 2003.

[12] Grid Interoperability Now Community Group (GIN-CG). Group's website. http://forge.ogf.org/sf/projects/gin, 2006.

[13] L. Grit, D. Inwin, A. Yumerefendi, and J. Chase. Virtual machine hosting for networked clusters: Building the foundations for 'autonomic' orchestration. In *1st International Workshop on Virtualization Technology in Distributed Computing (VTDC 2006)*, Tampa, USA, November 2006.

[14] A. Iosup, D. H. J. Epema, T. Tannenbaum, M. Farrellee, and M. Livny. Inter-operating Grids through delegated matchmaking. In *2007 ACM/IEEE Conference on Supercomputing (SC 2007)*, Reno, USA, November 2007.

[15] D. Irwin, J. Chase, L. Grit, A. Yumerefendi, D. Becker, and K. G. Yocum. Sharing networked resources with brokered leases. In *USENIX Annual Technical Conference*, pages 199–212, Boston, USA, June 2006.

[16] M. Islam, P. Balaji, P. Sadayappan, and D. K. Panda. QoPS: A QoS based scheme for parallel job scheduling. In *JSSPP 2003*, LNCS, pages 252–268, Seattle, USA, 2003.

[17] B. G. Lawson and E. Smirni. Multiple-queue backfilling scheduling with priorities and reservations for parallel systems. In *JSSPP 2002*, LNCS, pages 72–87, London, UK, 2002.

[18] C. Metz. Interconnecting ISP networks. *IEEE Internet Computing*, 5(2):74–80, 2001.

[19] K. Miura. Overview of Japanese science Grid project NAREGI. *Progress in Informatics*, pages 67–75, 2006.

[20] OneLab2. Project website. http://www.one-lab-2.org/, 2007.

[21] Open Science Grid. Project website. http://www.opensciencegrid.org, 2005.

[22] L. Peterson, S. Muir, T. Roscoe, and A. Klingaman. PlanetLab architecture: An overview. Technical Report PDN-06-031, PlanetLab Consortium, Princeton, USA, May 2006.

[23] L. Peterson and J. Wroclawski. Overview of the GENI architecture. GENI Design Document GDD-06-11, GENI: Global Environment for Network Innovations, January 2007.

[24] L. Ramakrishnan, D. Irwin, L. Grit, A. Yumerefendi, A. Iamnitchi, and J. Chase. Toward a doctrine of containment: Grid hosting with adaptive resource control. In *2006 ACM/IEEE Conference on Supercomputing (SC 2006)*, page 101, New York, USA, 2006.

[25] R. Ranjan, A. Harwood, and R. Buyya. SLA-based coordinated superscheduling scheme for computational Grids. In *IEEE International Conference on Cluster Computing (Cluster 2006)*, pages 1–8, Barcelona, Spain, September 2006.

[26] S. Surana, B. Godfrey, K. Lakshminarayanan, R. Karp, and I. Stoica. Load balancing in dynamic structured peer-to-peer systems. *Performance Evaluation*, 63(3):217–240, 2006.

[27] The Distributed ASCI Supercomputer 2 (DAS-2). Dutch University Backbone, 2006.

[28] Y. Wang, D. Scardaci, B. Yan, and Y. Huang. Interconnect EGEE and CNGRID e-infrastructures through interoperability between gLite and GOS middlewares. In *International Grid Interoperability and Interoperation Workshop (IGIIW 2007)*, pages 553–560, Bangalore, India, 2007.

[29] Y.-T. Wang and R. J. T. Morris. Load sharing in distributed systems. *IEEE Transactions on Computers*, C-34(3):204–217, March 1985.