Available online at www.sciencedirect.com
ScienceDirect
Future Generation Computer Systems 24 (2008) 280–295
FGCS
FUTURE GENERATION COMPUTER SYSTEMS
www.elsevier.com/locate/fgcs

A case for cooperative and incentive-based federation of distributed clusters☆

Rajiv Ranjan*, Aaron Harwood, Rajkumar Buyya

GRIDS Lab and P2P Group, Department of Computer Science and Software Engineering, University of Melbourne, Victoria, Australia

Received 18 April 2006; received in revised form 22 May 2007; accepted 22 May 2007
Available online 15 June 2007

Abstract

Research interest in Grid computing has grown significantly over the past five years. Management of distributed resources is one of the key issues in Grid computing. Central to management of resources is the effectiveness of resource allocation as it determines the overall utility of the system. The current approaches to brokering in a Grid environment are non-coordinated since application-level schedulers or brokers make scheduling decisions independently of the others in the system. Clearly, this can exacerbate the load sharing and utilization problems of distributed resources due to sub-optimal schedules that are likely to occur. To overcome these limitations, we propose a mechanism for coordinated sharing of distributed clusters based on computational economy. The resulting environment, called *Grid-Federation*, allows the transparent use of resources from the federation when local resources are insufficient to meet its users' requirements. The use of computational economy methodology in coordinating resource allocation not only facilitates the Quality of Service (QoS)-based scheduling, but also enhances utility delivered by resources. We show by simulation, while some users that are local to popular resources can experience higher cost and/or longer delays, the overall users' QoS demands across the federation are better met. Also, the federation's average case message-passing complexity is seen to be scalable, though some jobs in the system may lead to large numbers of messages before being scheduled.

1. Introduction

Clusters of computers have emerged as mainstream parallel and distributed platforms for high-performance, high-throughput and high-availability computing. Grid [19] computing extends the cluster computing idea to wide-area networks. A grid consists of cluster resources that are usually distributed over multiple administrative domains, managed and owned by different organizations having different resource management policies. With the large scale growth of networks and their connectivity, it is possible to couple these cluster resources as a part of one large Grid system. Such large scale resource coupling and application management is a complex undertaking, as it introduces a number of challenges in the domain of security, resource/policy heterogeneity, resource discovery, fault tolerance, dynamic resource availability and underlying network conditions.

The resources on a Grid (e.g. clusters, supercomputers) are managed by local resource management systems (LRMSes) such as Condor [28] and PBS [7]. These resources can also be loosely coupled to form campus grids using multi-clustering systems such as SGE [22] and LSF [40] that allow sharing of clusters owned by the same organization. In other words, these systems do not allow their combination similar to autonomous systems, to create an environment for *cooperative federation* of clusters, which we refer as Grid-Federation.

Other related concept called Virtual Organization (VO) [19]-based Grid resource sharing has been proposed in the literature. Effectively, a VO is formed to solve specific scientific problems. All the participants follow the same resource management policies defined by a VO. Hence, a VO represents a socialist world, wherein the participants have to adhere to community-wide agreed policies and priorities. In contrast, proposed Grid-Federation is a democratic world with complete autonomy for each participant. Further, a participant in the federation can behave rationally as we propose the use of economic model for resource management. Grid-Federation users submit their job to the local scheduler. In case local resources are not available or are not able to meet the requirement then job is transparently migrated to a remote resource (site) in the federation, although

☆ This is an extended version of paper that was published with Cluster'05, Boston, MA.

* Corresponding author.
E-mail addresses: rranjan@cs.mu.oz.au, rranjan@csse.unimelb.edu.au (R. Ranjan), aharwood@cs.mu.oz.au (A. Harwood), raj@cs.mu.oz.au (R. Buyya).

0167-739X/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2007.05.006

this job migration is driven by the users' QoS requirements. In a VO, user jobs are managed by a global scheduler which enforces resource allocation based on VO-wide policies.

Scheduling jobs across resources that belong to distinct administrative domains is referred to as *superscheduling*. The majority of existing approaches to superscheduling [32] in a Grid environment is non-coordinated. Superschedulers or resource brokers such as Nimrod-G [1], Tycoon [27], and Condor-G [21] perform scheduling related activities independent of the other superschedulers in the system. They directly submit their applications to the underlying resources *without* taking into account the current load, priorities, utilization scenarios of other application-level schedulers. Clearly, this can lead to over-utilization or a bottleneck on some valuable resources while leaving others largely underutilized. Furthermore, these superschedulers do not have a coordination mechanism and this exacerbates the load sharing and utilization problems of distributed resources because sub-optimal schedules are likely to occur.

Furthermore, end-users or their application-level superschedulers submit jobs to the LRMS without having knowledge about response time or service utility. Sometimes these jobs are queued for relatively excessive times before being actually processed, leading to degraded QoS. To mitigate such long processing delays and to enhance the value of computation, a scheduling strategy can use priorities from competing user jobs that indicate varying levels of importance. This is a widely studied scheduling technique (e.g. using priority queues) [3]. To be effective, the schedulers require knowledge of how users value their computations in terms of QoS requirements, which usually varies from job to job. LRMS schedulers can provide a feedback signal that prevents the user from submitting unbounded amounts of work.

Currently, system-centric approaches such as Legion [13, 38], NASA-Superscheduler [33], Condor, Condor-Flock [8], Apples [6], PBS and SGE provide limited support for QoS driven resource sharing. These system-centric schedulers, allocate resources based on parameters that enhance system utilization or throughput. The scheduler either focuses on minimizing the response time (sum of queue time and actual execution time) or maximizing overall resource utilization of the system and these are not specifically applied on a per-user basis (user oblivious). System-centric schedulers treat all resources with the same scale, as if they are worth the same and the results of different applications have the same value; while in reality the resource provider may value his resources differently and has a different objective function. Similarly, a resource consumer may value various resources differently and may want to negotiate a particular price for using a resource. Hence, resource consumers are unable to express their valuation of resources and QoS parameters. Furthermore, the system-centric schedulers do not provide any mechanism for resource owners to define what is shared, who is given the access and the conditions under which sharing occurs [20].

1.1. Grid-Federation

To overcome these shortcomings of non-coordinated, system-centric scheduling systems, we propose a new distributed resource management model, called Grid-Federation. Our Grid-Federation system is defined as a large scale resource sharing system that consists of a coordinated federation (the term is also used in the Legion system and should not be confused with our definition), of distributed clusters based on policies defined by their owners (shown in Fig. 1). Fig. 1 shows an abstract model of our Grid-Federation over a shared federation directory. To enable policy-based transparent resource sharing between these clusters, we define and model a new RMS system, which we call Grid-Federation Agent (GFA). Currently, we assume that the directory information is shared using some efficient protocol (e.g. a peer-to-peer protocol [29, 25]). In this case the P2P system provides a decentralized database with efficient updates and range query capabilities. Individual GFAs access the directory information using the interfaces shown in Fig. 1, i.e. subscribe, quote, unsubscribe, query. In this paper, we are not concerned with the specifics of the interface (which can be found in [30]) although we do consider the implications of the required message-passing, i.e. the messages sent between GFAs to undertake the scheduling work.

Our approach considers the emerging computational economy metaphor [1,36,37] for Grid-Federation. In this case resource owners: can clearly define what is shared in the Grid-Federation while maintaining complete autonomy; can dictate who is given access; and receive incentives for leasing their resources to federation users. We adopt the market-based economic model from [1] for resource allocation in our proposed framework. Some of the commonly used economic models [9] in resource allocation include the commodity market model, the posted price model, the bargaining model, the tendering/contract-net model, the auction model, the bid-based proportional resource sharing model, the community/coalition model and the monopoly model. We focus on the commodity market model [39]. In this model every resource has a price, which is based on the demand, supply and value in the Grid-Federation. Our Economy model driven resource allocation methodology focuses on: (i) optimizing resource provider's objective functions, (ii) increasing end-user's perceived QoS value based on QoS level indicators [30] and QoS constraints.

The key contribution of the paper includes our proposed new distributed resource management model, called Grid-Federation, which provides: (i) a market-based Grid superscheduling technique; (ii) decentralization via a shared federation directory that gives site autonomy and scalability; (iii) ability to provide admission control facility at each site in the federation; (iv) incentives for resources owners to share their resources as part of the federation; and (v) access to a larger pool of resources for all users. In this paper, we demonstrate, by simulation, the feasibility and effectiveness of our proposed Grid-Federation.

The rest of the paper is organized as follows. Section 2 explores various related projects. In Section 3 we summarize our Grid-Federation and Section 4 deals with various experiments that we conducted to demonstrate the utility of our work. We end the paper with some concluding remarks and future work in Section 5.

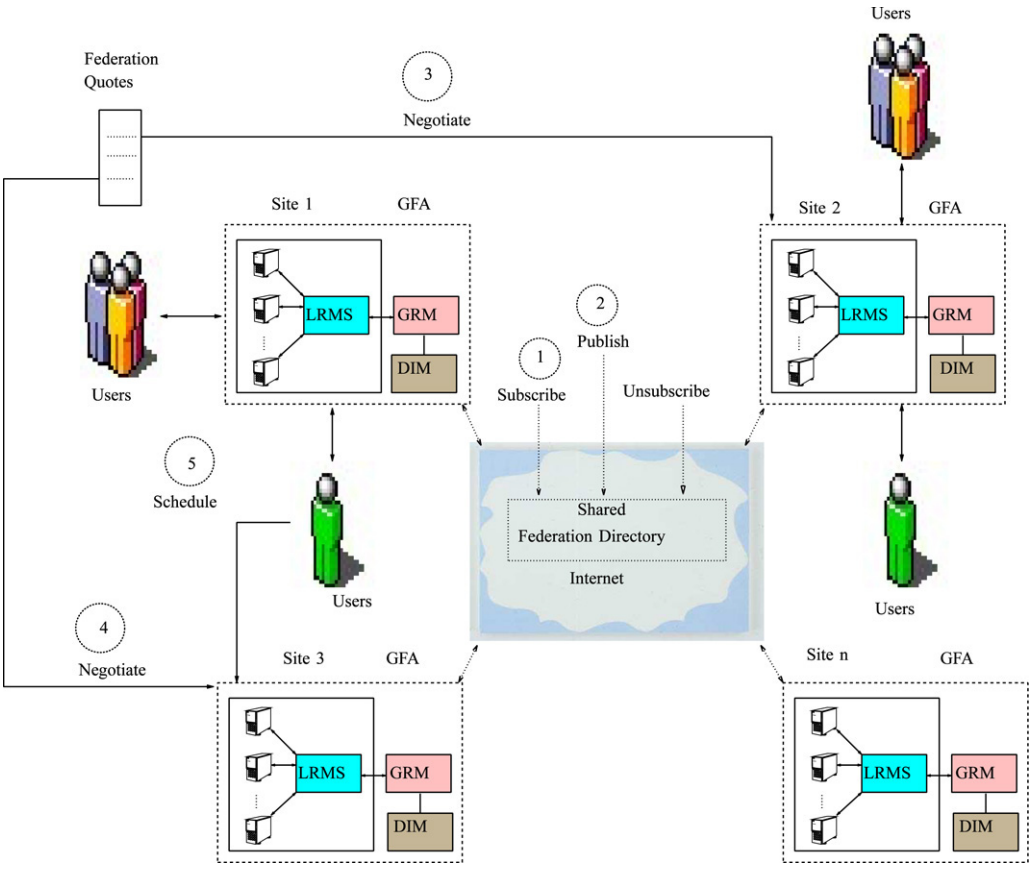


Fig. 1. Grid-Federation resource sharing system. Grid-Federation consists of cluster resources that are distributed over multiple ogranisations and control domains. Each site in the federation instantiates a GFA resource management system that exports the local resources to the federation. GFA undertakes one-to-one negotiation as part of job superscheduling process.

2. Related work

Resource management and scheduling for parallel and distributed systems has been investigated extensively in the recent past (AppLes, NetSolve [12], Condor, LSF, SGE, Legion, Condor-Flock, NASA-Superscheduler, Nimrod-G and Condor-G). In this paper, we mainly focus on superscheduling systems that allow scheduling of jobs across wide-area distributed clusters. We highlight the current scheduling methodology followed by Grid superscheduling systems including NASA-Superscheduler, Condor-Flock (based on P2P substrate Pastry [31]), Legion-based federation and Resource Brokers. Furthermore, we also discuss some of the computational economy-based Cluster and Grid systems.

The work in [33] models a Grid superscheduler architecture and presents three different distributed job migration algorithms. In contrast to this superscheduling system, our approach differs in the following (i) the job-migration or the load-balancing in the Grid-Federation is driven by user specified QoS constraints and resource owners' sharing policies; (ii) our approach gives a resource owner complete autonomy over resource allocation decision; and (iii) our superscheduling mechanism utilizes decentralized shared federation directory for indexing and querying the resources.

The work in [8] presents a superscheduling system that consists of Internet-wide Condor work pools. They utilize Pastry routing substrate to organize and index the Condor work pool. The superscheduling mechanism is based on system-centric parameters. In comparison with this work, Grid-Federation is based on decentralized shared federation directory. Further, our superscheduling scheme considers user-centric parameters for job scheduling across the federation.

OurGrid [4] provides a Grid superscheduling middleware infrastructure based on the P2P network paradigm. The OurGrid community is basically a collection of a number of OurGrid Peer (OG Peer) that communicate using P2P protocols. Superscheduling in OurGrid is primarily driven by the site's reputation in the community. In contrast, we propose a more generalized resource sharing system based on real-market models. Further, our superscheduling system focuses on optimizing resource owners' and consumers' objective functions.

Bellagio [5] is a market-based resource allocation system for federated distributed computing infrastructures. Resource allocation in this system is based on bid-based proportional resource sharing model. Bids for resources are cleared by a centralized auctioneer. In contrast, we propose a decentralized superscheduling system based on commodities markets. Resource allocation decision in our proposed system is controlled by the concerned site, hence providing complete site autonomy.

Table 1
Superscheduling technique comparison

Index	System name	Network model	Scheduling parameters	Scheduling mechanism
1	NASA-Superscheduler	Random	System-centric	Partially coordinated
2	Condor-Flock P2P	P2P (Pastry)	System-centric	Partially coordinated
3	Grid-Federation	P2P (Decentralized directory)	User-centric	Coordinated
4	Legion-Federation	Random	System-centric	Coordinated
5	Nimrod-G	Centralized	User-centric	Non-coordinated
6	Condor-G	Centralized	System-centric	Non-coordinated
7	Our-Grid	P2P	System-centric	Coordinated
8	Tycoon	Centralized	User-centric	Non-coordinated
9	Bellagio	Centralized	User-centric	Coordinated

Tycoon [27] is a distributed market-based resource allocation system. Application scheduling and resource allocation in Tycoon is based on decentralized isolated auction mechanism. Every resource owner in the system runs its own auction for his local resources. Furthermore, auctions are held independently, thus clearly lacking any coordination. In contrast, we propose a mechanism for cooperative and coordinated sharing of distributed clusters based on computational economy. We apply commodity market model for regulating the supply and demand of resources in the Grid-Federation.

Nimrod-G [1] is a resource management system (RMS) that serves as a resource broker and supports deadline and budget constrained algorithms for scheduling task-farming applications on the platform. The superscheduling mechanism inside the Nimrod-G does not take into account other brokering systems currently present in the system. This can lead to over-utilization of some resources while under-utilization of others. To overcome this, we propose a set of distributed brokers having a transparent coordination mechanism.

Other systems including Libra [34] and REXEC [16] apply market methodologies for managing cluster resources within a single administrative domain. Finally in Table 1, we summarize various superscheduling systems based on underlying network model, scheduling parameter and scheduling mechanism.

3. Grid-Federation: Architecture for decentralized resource management

(1) *Grid-Federation agent*: We define our Grid-Federation (shown in Fig. 1) as a mechanism that enables logical coupling of cluster resources. The Grid-Federation supports policy-based [14] transparent sharing of resources and QoS-based [26] job scheduling. We also propose a new computational economy metaphor for cooperative federation of clusters. Computational economy [1,36,37] enables the regulation of supply and demand of resources, offers incentive to the resource owners for leasing, and promotes QoS-based resource allocation. The Grid-Federation consists of the cluster owners as resource providers and the end-users as resource consumers. End-users are also likely to be topologically distributed, having different performance goals, objectives, strategies and demand patterns. We focus on optimizing the resource provider's objective and resource consumer's utility functions by using a quoting mechanism. The Grid-Federation consists of cluster resources distributed across multiple organizations and administrative domains. To enable policy-based coordinated resource sharing between these clusters, we define and model a new RMS system, which we call Grid-Federation Agent (GFA). A cluster can become a member of the federation by instantiating a GFA component. GFA acts as a resource coordinator in the federated space, spanning over all the clusters. These GFAs in the federation inter-operate using an agreed communication primitive over the shared federation directory.

This section provides comprehensive details about our proposed Grid-Federation, including models used for budget and deadline calculations in the simulations of the next section. The model defines the following functional modules of a GFA: Grid Resource Manager (**GRM**)

The Grid resource manager is responsible for superscheduling the locally submitted jobs in the federation. Further, it also manages the execution of remote jobs in conjunction with the LRMS on the local resource. *Local jobs* refers to the jobs submitted by the local population of users, while *remote jobs* refers to the incoming jobs from remote GRMs. A GRM provides admission control facility at each site in the federation. Fig. 2 shows the Grid-Federation superscheduling architecture that we propose. In Fig. 2, a GFA i in the federation with modules GRM, LRMS and DIM is shown. The GRM component of GFA is connected to the federation queue which accepts the incoming remote jobs (from the federation) as well as local jobs. All the remote jobs are transferred to the local queue which is controlled by the GFA's LRMS module. A GRM can also export the locally submitted jobs to other sites in the federation depending on the user specified QoS requirements. The job submission and migration process is represented by a dashed arrow in Fig. 2.

A local user submits his job to the GRM which then places it in the federation queue. GRM analyses the user's QoS specification and then sends a query message to the DIM. The DIM returns the first fastest or first cheapest machine as specified in the QoS requirements. If the returned machine is the local resource then the job is transferred to the local queue. Otherwise, the job is transferred to a remote site in the federation. GRMs undertake one-to-one negotiation before submitting a job to a remote site. The GRM local to the submitted job sends admission control negotiate message to the remote GRM requesting a guarantee on the total job completion time. Following this, the contacted GRM queries its LRMS. If the LRMS reports that the job can be completed within

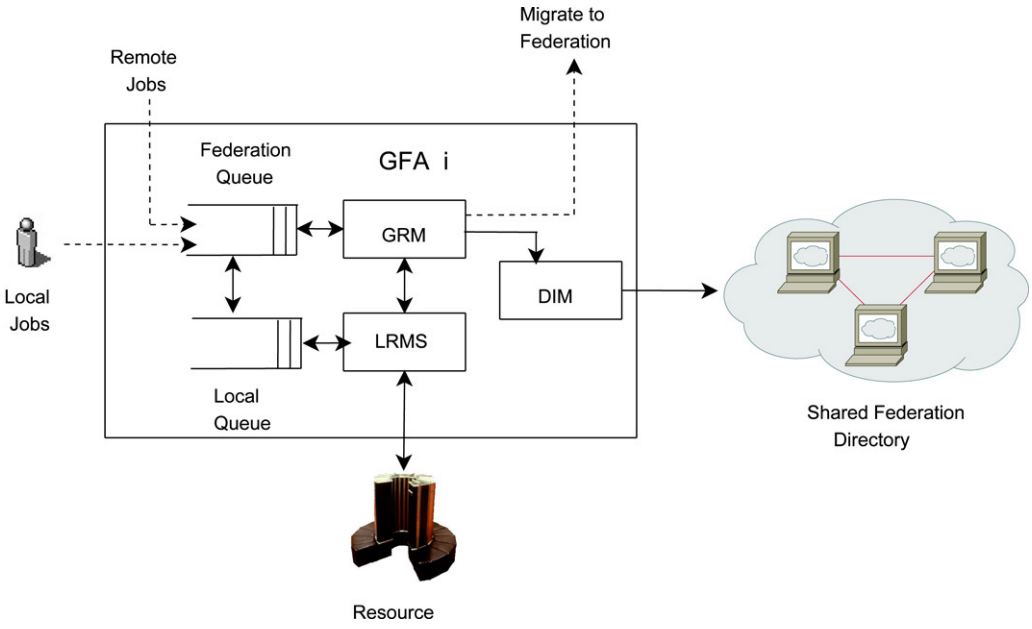


Fig. 2. Grid-Federation superscheduling architecture: A GFA i in the federation with modules GRM, LRMS and DIM. GRM is connected to a federation queue which accepts incoming remote jobs (from federation) and local jobs. All remote jobs are transferred to the local queue which is controlled by the LRMS module. A GRM can also migrate the locally submitted jobs to other sites in the federation depending on the user specified QoS requirements. A job submission and migration process is represented by a dashed arrow in the figure.

the specified deadline, then the admission control acceptance message is sent to the requesting GRM. On receiving the acceptance, the GRM sends the job. The inter-site GRM-to-GRM negotiation scheme prevents the GRMs from submitting unlimited amount of jobs to the resources. Further, this approach allows autonomy for every resource domain, as they have the capability to perform per-job basis admission control decision. All migrated jobs are first queued in the federation queue, which are then subsequently transferred to the local queue for final execution process.

The proposed Grid-Federation mechanism can leverage services of Grid-Bank [2] for credit management. The participants in the system can use Grid-Bank to exchange Grid Dollars.

Local Resource Management System (**LRMS**)

In our proposed Grid-Federation distributed resource sharing system, we assume that every cluster has a generalized RMS, such as a SGE or PBS that manages cluster wide resource allocation and application scheduling. Most of the available RMS packages have a centralised organisation similar to the master-worker pool model. In the centralised organisation, there is only one scheduling controller (master node) which coordinates system-wide decisions. Grid resource manager queries LRMS to acquire information about local job queue size, expected response time for a job, and resource utilisation status.

Distributed Information Manager (**DIM**)

The DIM performs tasks like resource discovery and advertisement through well defined primitives. It interacts with an underlying shared federation directory (shown in Fig. 2). Recall that we assume the directory information is shared using some efficient protocol (e.g. a P2P protocol). In this case, the P2P system provides a decentralized database with efficient updates and range query capabilities. Individual GFAs access the directory information using the interface shown in Fig. 1, i.e. subscribe, quote, unsubscribe and query. In this paper, we are not concerned with the specifics of the interface (which can be found in [30]). The resource discovery function includes searching for suitable cluster resources while resource advertisement is concerned with advertising resource capability (with pricing policy) to other clusters in the federation. The federation directory maintains quotes or advertised costs from each GFA in the federation. Each quote consists of a resource description R_i, for cluster i, and a cost c_i for using that resource configured by respective cluster owners. Using R_i and c_i, a GFA can determine the cost of executing a job on cluster i and the time taken, assuming that the cluster i has no load. The actual load of the cluster needs to be determined dynamically and the load can lead to changes in time taken for job completion. In this work, we assume that c_i remains static throughout the simulations. Each GFA can query the federation directory to find the kth fastest cluster or the kth cheapest cluster. We assume the query process is optimal, i.e. that it takes $O(\log n)$ messages [11] to query the directory, when there are n GFAs in the system, we consider the number of additional messages that are used to satisfy our Grid-Federation scheduling process.

3.1. Decentralised market place and Grid-Federation

Grid computing assembles resources that are well managed, powerful and well connected to the Internet. Grids present a platform for Grid participants (GPs) to collaborate and coordinate resource management activities. Key GPs include the *producers* (Grid resource-owners) and *consumers* (Grid users). GPs have different goals, objectives, strategies, and

supply and demand functions. GPs are topologically distributed and belong to different administrative domains. Controlled administration of Grid resources gives an ability to provide a desired QoS in terms of computational and storage efficiency, software or library upgrades. However, such controlled administration of resources gives rise to various social and political issues on which these resources are made available to the outside world.

A resource owner invests a significant amount of money in establishing the resource such as, initial cost of buying, setting up, maintenance cost including hiring the administrator and expense of timely software and the hardware upgrades. There is a complex diversity in terms of resources' usage policies, loads and availability. Resource owners in a grid behave as rational participants having distinct stake holdings with potentially conflicting and diverse utility functions. In this case, resource owners apply resource sharing policies that tend to maximize their utility functions [17,23]. Similarly, the resource consumers in a grid associate QoS-based utility constraints with their applications and expect that the constraints are satisfied within the acceptable limits. Every resource owner makes the policy related decision independently that best optimizes his objective function. Likewise, resource consumers have diverse QoS-based utility constraints, priorities and demand patterns.

To capture the above dynamics and complexity of Grid resource sharing environment, Grid-Federation applies market-based economy principles for resource allocation and application scheduling. In particular, we adopt commodity market model. In this model, every resource owner sets up a fixed price based on the demand for his resources in the decentralised market place. The resource owner advertises its resource access cost through its local GFA service. Analyzing different pricing algorithm based on supply and demand function is a vast research area. Investigating how the cluster owners determine the price [15,35,39] of their commodity is a subject of future work.

3.2. General Grid-Federation superscheduling technique

In this section we describe our general Grid-Federation scheduling technique. In Fig. 1 a user who is local to GFA 1 is submitting a job. If the user's job QoS cannot be satisfied locally then GFA 1 queries the federation directory to obtain the quote of the first fastest or first cheapest cluster. In this case, the federation directory returns the quote advertised by GFA 2. Following this, GFA 1 sends a negotiate message (enquiry about QoS guarantee in terms of response time) to GFA 2. If GFA has too much load and cannot complete the job within the deadline then GFA 1 queries the federation directory for the second cheapest/fastest GFA and so on. The query-negotiate process is repeated until GFA 3 finds a GFA that can schedule the job (in this example the job is finally scheduled on site 3).

Every federation user must express how much he is willing to pay, called a *budget*, and required response time, called a *deadline*, for his job number j. In this work, we say that a job's QoS has been satisfied if the job is completed within budget and deadline, otherwise it is not satisfied. Every cluster in the federation has its own resource set R_i which contains the definition of all resources owned by the cluster and ready to be offered. R_i can include information about the CPU architecture, number of processors, RAM size, secondary storage size, operating system type, etc. In this work, $R_i = (p_i, \mu_i, \gamma_i)$ which includes the number of processors, p_i, their speed, μ_i and underlying interconnect network bandwidth γ_i. We assume that there is always enough RAM and correct operating system conditions, etc. The cluster owner charges c_i per unit time or per unit of million instructions (MI) executed, e.g. per 1000 MI.

We write $J_{i,j,k}$ to represent the ith job from the jth user of the kth resource. A job consists of the number of processors required, $p_{i,j,k}$, the job length, $l_{i,j,k}$ (in terms of instructions), the budget, $b_{i,j,k}$, the deadline or maximum delay, $d_{i,j,k}$ and the communication overhead, $\alpha_{i,j,k}$.

To capture the nature of parallel execution with message-passing overhead involved in the real application, we considered a part of total execution time as the communication overhead and remaining as the computational time. In this work, we consider the network communication overhead $\alpha_{i,j,k}$ for a parallel job $J_{i,j,k}$ to be randomly distributed over the processes. In other words, we do not consider the case e.g. when a parallel program written for a hypercube is mapped to a mesh architecture. We assume that the communication overhead parameter $\alpha_{i,j,k}$ would scale the same way over all the clusters depending on γ_i. The total data transfer involved during a parallel job execution is given by

$$\Gamma(J_{i,j,k}, R_k) = \alpha_{i,j,k}\gamma_k. \tag{1}$$

The time for job $J_{i,j,k} = (p_{i,j,k}, l_{i,j,k}, b_{i,j,k}, d_{i,j,k}, \alpha_{i,j,k})$ to execute on resource R_m is

$$D(J_{i,j,k}, R_m) = \frac{l_{i,j,k}}{\mu_m p_{i,j,k}} + \frac{\Gamma(J_{i,j,k}, R_k)}{\gamma_m} \tag{2}$$

$$D(J_{i,j,k}, R_m) = \frac{l_{i,j,k}}{\mu_m p_{i,j,k}} + \frac{\alpha_{i,j,k}\,\gamma_k}{\gamma_m} \tag{3}$$

and the associated cost is

$$B(J_{i,j,k}, R_m) = c_m \frac{l_{i,j,k}}{\mu_m p_{i,j,k}}. \tag{4}$$

If $s_{i,j,k}$ is the time that $J_{i,j,k}$ is submitted to the system then the job must be completed by the time $s_{i,j,k} + d_{i,j,k}$.

3.3. QoS driven resource allocation algorithm for Grid-Federation

We consider a deadline and budget constrained (DBC) scheduling algorithm, or cost–time optimization scheduling. The federation user can specify any one of the following optimization strategies for their jobs:

- optimization for time (OFT) — give minimum possible response time within the budget limit;
- optimization for cost (OFC) — give minimum possible cost within the deadline.

For each job that arrives at a GFA, called the local GFA, the following is done:

(1) Set $r = 1$.
(2) If OFT is required for the job then query the federation directory for the rth fastest GFA; otherwise OFC is required and the query is made for the rth cheapest GFA. Refer to the result of the query as the remote GFA.
(3) The local GFA sends a message to the remote GFA, requesting a guarantee on the time to complete the job.
(4) If the remote GFA confirms the guarantee then the job is sent, otherwise $r := r + 1$ and the process iterates through step (2).

Recall that we assume each query takes $O(\log n)$ messages and hence in this work we use simulation to study how many times the iteration is undertaken, on a per job basis and on a per GFA basis. The remote GFA makes a decision immediately upon receiving a request as to whether it can accept the job or not. If the job's QoS parameters cannot be satisfied (after iterating up to the greatest r such that GFA could feasibly complete the job) then the job is dropped.

Effectively, for job $J_{i,j,k}$ that requires OFC then GFA m with R_m is chosen such that $B(J_{i,j,k}, R_m) = \min_{1<m'\leq n}\{B(J_{i,j,k}, R_{m'})\}$, and $D(J_{i,j,k}, R_m) \leq s_{i,j,k} + d_{i,j,k}$. Similarly, for OFT then GFA m is chosen such that $D(J_{i,j,k}, R_m) = \min_{1<m'\leq n}\{D(J_{i,j,k}, R_{m'})\}$, and $B(J_{i,j,k}, R_m) \leq b_{i,j,k}$.

3.4. Quote value

We assume that c_i remains static throughout the simulations. In this work, we are only interested in studying the effectiveness of our Grid-Federation superscheduling algorithm based on the static access charge c_i. In simulations, we configure c_i using the function:

$$c_i = f(\mu_i) \tag{5}$$

where,

$$f(\mu_i) = \frac{c}{\mu}\,\mu_i \tag{6}$$

c is the access price and μ is the speed of the fastest resource in the Grid-Federation.

3.5. User budget and deadline

While our simulations in the next section use trace data for job characteristics, the trace data does not include user specified budgets and deadlines on a per-job basis. In this case we are forced to fabricate these quantities and we include the models here.

For a user, j, we allow each job from that user to be given a budget (using Eq. (4)),

$$b_{i,j,k} = 2\,B(J_{i,j,k}, R_k). \tag{7}$$

In other words, the total budget of a user over simulation is unbounded and we are interested in computing the budget that is required to schedule all of the jobs.

Also, we let the deadline for job i (using Eq. (2)) be

$$d_{i,j,k} = 2\,D(J_{i,j,k}, R_k). \tag{8}$$

We assign two times the value of total budget and deadline for the given job, as compared to the expected budget spent and the response time on the originating resource.

4. Experiments and analysis

4.1. Workload and resource methodology

We used trace-based simulation to evaluate the effectiveness of the proposed system and the QoS provided by the proposed superscheduling algorithm. The workload trace data was obtained from [18]. The trace contains the real time workload of various supercomputers/resources that are deployed at the Cornell Theory Center (CTC SP2), Swedish Royal Institute of Technology (KTH SP2), Los Alamos National Lab (LANL CM5), LANL Origin 2000 Cluster (Nirvana) (LANL Origin), NASA Ames (NASA iPSC) and San-Diego Supercomputer Center (SDSC Par96, SDSC Blue, SDSC SP2) (See Table 2). The workload trace is a record of usage data for parallel jobs that were submitted to various resource facilities. Every job arrives, is allocated one or more processors for a period of time, and then leaves the system. Furthermore, every job in the workload has an associated arrival time, indicating when it was submitted to the scheduler for consideration. As the experimental trace data does not include details about the network communication overhead involved for different jobs, we artificially introduced the communication overhead element as 10% of the total parallel job execution time.

The simulator was implemented using the GridSim [10] toolkit that allows modeling and simulation of distributed system entities for evaluation of scheduling algorithms. GridSim offers a concrete base framework for simulation of different kinds of heterogeneous resources, brokering systems and application types. This toolkit can simulate resource brokering for resources that belong to a single administrative domain (such as a cluster) or multiple administrative domain (such as a grid). The core of simulation is based on *simjava* [24], a discrete event simulation package implemented in Java. The main classes of GridSim include GridResource, GridSim, Gridlet, AllocPolicy and GridInformationService. These classes communicate using discrete message-passing events. To enable parallel workload simulation with GridSim, we extend the existing AllocPolicy and SpaceShared entities.

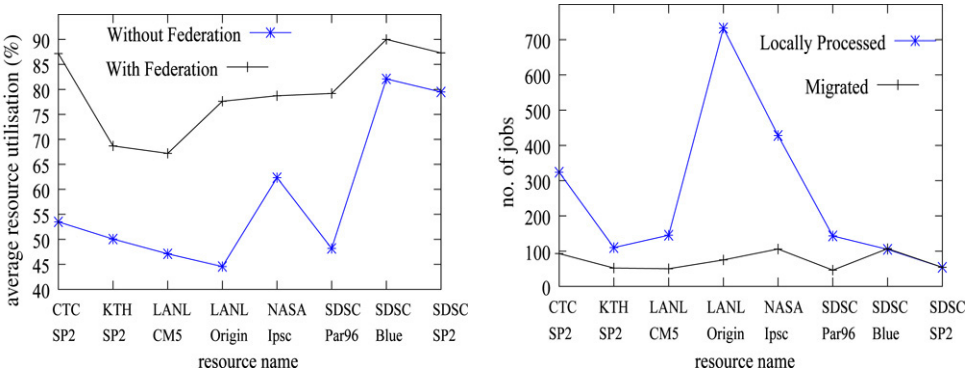
Our simulation environment models the following basic entities in addition to existing entities in GridSim:

- local user population — models the workload obtained from trace data;
- GFA — generalized RMS system;
- GFA queues (federation and local) — placeholder for incoming jobs from local user population and the federation;
- GFA shared federation directory — simulates an efficient distributed query process such as peer-to-peer.

For evaluating the QoS driven resource allocation algorithm, we assigned a synthetic QoS specification to each resource including the Quote value (price that a cluster owner charges for service), having varying MIPS rating and underlying network

Table 2
Workload and resource configuration

Index	Resource/Cluster name	Trace date	Processors	MIPS (rating)	Total jobs in trace	Quote (price)	NIC to network bandwidth (Gb/s)
1	CTC SP2	June96–May97	512	850	79,302	4.84	2
2	KTH SP2	Sept96–Aug97	100	900	28,490	5.12	1.6
3	LANL CM5	Oct94–Sept96	1024	700	201,387	3.98	1
4	LANL Origin	Nov99–Apr2000	2048	630	121,989	3.59	1.6
5	NASA iPSC	Oct93–Dec93	128	930	42,264	5.3	4
6	SDSC Par96	Dec95–Dec96	416	710	38,719	4.04	1
7	SDSC Blue	Apr2000–Jan2003	1152	730	250,440	4.16	2
8	SDSC SP2	Apr98–Apr2000	128	920	73,496	5.24	4



(a) Average resource utilization (%) vs. resource name. (b) No. of jobs vs. resource name.

Fig. 3. Resource utilization and job migration plot.

communication bandwidth. The simulation experiments were conducted by utilizing workload trace data over the total period of 2 days (in simulation units) at all the resources. Hence, effectively our simulation considers only a fraction of jobs per computing site as compared to the total number of jobs that were submitted. For example, originally 79,302 jobs were submitted to CTC SP2 over a period of 1 year, while our simulation considered only 417 jobs (no. of jobs submitted over 2 days). We consider the following resource sharing environment for our experiments:

- independent resource — Experiment 1;
- federation without economy — Experiment 2;
- federation with economy — Experiments 3, 4 and 5.

4.2. Experiment 1 — independent resources

In this experiment the resources were modeled as an independent entity (without federation). All the workload submitted to a resource is processed and executed locally (if possible). In Experiment 1 (and 2) we consider, if the user request cannot be served within a requested deadline, then it is rejected otherwise it is accepted. In original trace, as jobs were supposed to be scheduled on the local resource, they were queued in until the required number of processors became available. Effectively, no job was rejected in the original trace. During Experiment 1 (and 2), we evaluate the performance of a resource in terms of average resource utilization (amount of real work that a resource does over the simulation period excluding the queue processing and idle time), job acceptance rate (total percentage of jobs accepted) and conversely the job rejection rate (total percentage of jobs rejected). The result of this experiment can be found in Table 3 and Fig. 3. Experiment 1 is essentially the control experiment that is used as a benchmark for examining the effects of using federated (with and without economy) sharing of resources.

4.3. Experiment 2 — with federation

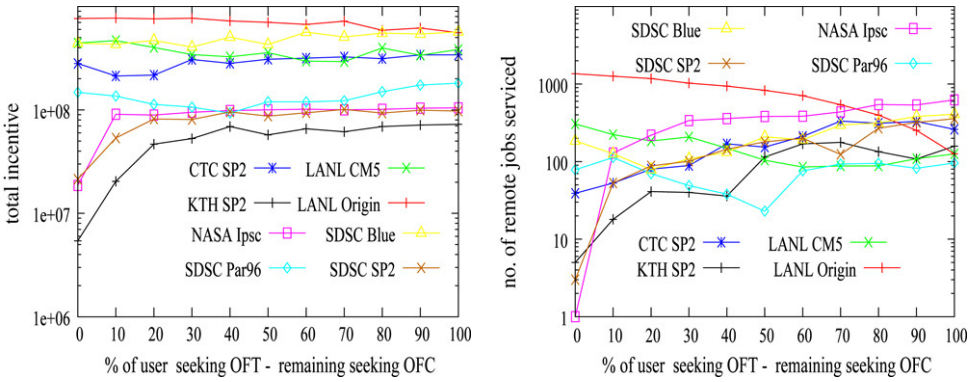
In this experiment, we analyzed the workload processing statistics of various resources when part of the Grid-Federation but not using an economic model. In this case the workload assigned to a resource can be processed locally. In case a local resource is not available then online scheduling is performed that considers the resources in the federation in decreasing order of their computational speed. We also quantify the jobs depending on whether they are processed locally or migrated to the federation. Table 4 and Fig. 3 describe the results of this experiment.

4.4. Experiment 3 — with federation and economy

In this experiment, we study the computational economy metaphor in the Grid-Federation. In order to study economy-based resource allocation mechanism, it was necessary to fabricate user budgets and job deadlines. As the trace data does not indicate these QoS parameters, we assigned them using Eqs. (7) and (8) to all the jobs across the resources. We performed the experiment under 11 different combinations of user population profile:

Table 3
Workload processing statistics (without federation)

Index	Resource/Cluster name	Average resource utilization (%)	Total jobs	Total jobs accepted (%)	Total jobs rejected (%)
1	CTC SP2	53.492	417	96.642	3.357
2	KTH SP2	50.06438	163	93.865	6.134
3	LANL CM5	47.103	215	83.72	16.27
4	LANL Origin	44.55013	817	93.757	6.24
5	NASA iPSC	62.347	535	100	0
6	SDSC Par96	48.17991	189	98.941	1.058
7	SDSC Blue	82.08857	215	57.67	42.3255
8	SDSC SP2	79.49243	111	50.45	49.54



(a) Total incentive (Grid Dollars) vs. user population profile.

(b) No. of remote jobs serviced vs. user population profile.

Fig. 4. Resource owner perspective.

$OFT = i$ and $OFC = 100 - i$ for $i = 0, 10, 20, \ldots, 100$. Figs. 4–8 describe the results of this experiment.

4.5. Experiment 4 — message complexity with respect to jobs

In this experiment, we consider the total incoming and outgoing messages at all GFAs. The various message types include negotiate, reply, job submission (messages containing actual job) and job completion (message containing job output). We quantify the number of local messages (sent from a GFA to undertake a local job scheduling) and remote messages (received at a GFA to schedule a job belonging to a remote GFA in the federation). The experiment was conducted for the same user population as explained in Experiment 3. Fig. 9 describes the result of this experiment.

4.6. Experiment 5 — message complexity with respect to system size

This experiment measures the system's performance in terms of the total message complexity involved as the system size grows from 10 to 50. In this case, we consider the average, max and min number of messages (sent/recv) per GFA/per job basis. Note that, in case n messages are undertaken to schedule a job then it involves traversing (if $n > 2$ then $(n-2)/2$, else $n/2$) the entries of the GFA list. To accomplish larger system size, we replicated our existing resources accordingly (shown in Table 1). The experiment was conducted for the same user population as explained in Experiment 3. Figs. 10 and 11 describe the results of this experiment.

4.7. Results and observations

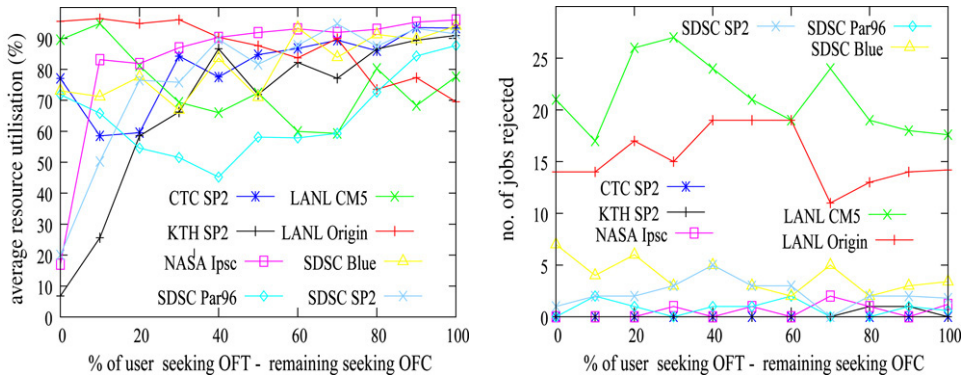
4.7.1. Justifying Grid-Federation-based resource sharing

During Experiment 1 we observed that 5 out of 8 resources remained underutilized (less than 60%). During Experiment 2, we observed that the overall resource utilization of most of the resources increased as compared to Experiment 1 (when they were not part of the federation), for instance resource utilization of CTC SP2 increased from 53.49% to 87.15%. The same trends can be observed for other resources too (refer to Fig. 3(a)). There was an interesting observation regarding migration of the jobs between the resources in the federation (load sharing). This characteristic was evident at all the resources including CTC SP2, KTH SP2, NASA iPSC etc. At CTC, which had a total of 417 jobs to schedule, we observed that 324 (refer to Table 4 or Fig. 3(b)) of them were executed locally while the remaining 93 jobs migrated and were executed at some remote resource in the federation. Further, CTC executed 72 remote jobs, which migrated from other resources in the federation.

The federation-based load sharing also leads to a decrease in the total job rejection rate, this can be observed in the case of resource SDSC Blue where the job rejection rate decreased from 42.32% to 1.39% (refer to Tables 3 and 4). Note that, the average job acceptance rate, over all resources in the federation, increased from 90.30% (without federation) to 98.61% (with federation). Thus, for the given job trace, it is preferable to make use of more resources, i.e. to migrate jobs. In other words,

Table 4
Workload processing statistics (with federation)

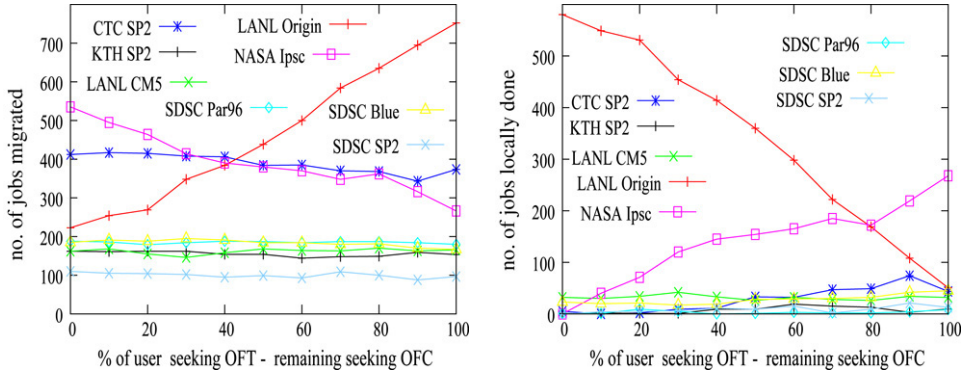
Index	Resource/Cluster name	Average resource utilization (%)	Total jobs	Total jobs accepted (%)	Total jobs rejected (%)	No. of jobs processed locally	No. of jobs migrated to federation	No. of remote jobs processed
1	CTC SP2	87.15	417	100	0	324	93	72
2	KTH SP2	68.69	163	99.38	0.61	110	52	35
3	LANL CM5	67.20	215	90.69	9.30	145	50	70
4	LANL Origin	77.62	817	98.89	1.10	733	75	81
5	NASA iPSC	78.73	535	99.81	0.18	428	106	129
6	SDSC Par96	79.17	189	100	0	143	46	30
7	SDSC Blue	90.009	215	98.60	1.39	105	107	77
8	SDSC SP2	87.285	111	97.29	2.70	54	54	89



(a) Average resource utilization (%) vs. user population profile.

(b) No. of jobs rejected vs. user population profile.

Fig. 5. Resource owner perspective.



(a) No. of jobs migrated vs. user population profile.

(b) No. of jobs locally done vs. user population profile.

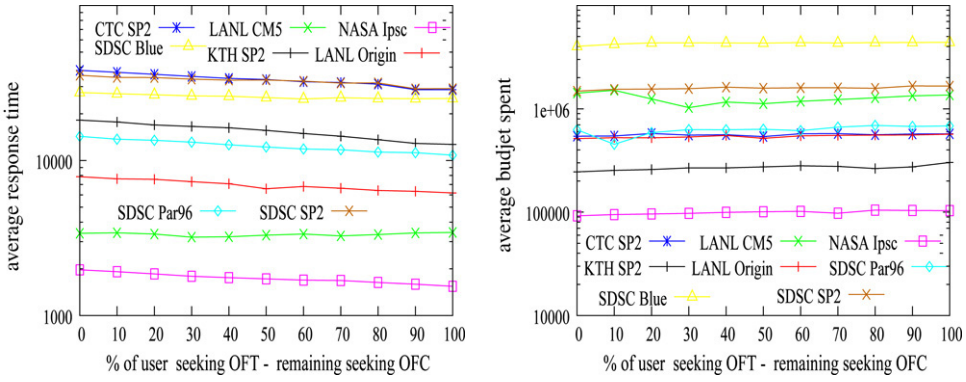
Fig. 6. Resource owner perspective.

the job trace shows the potential for resource sharing to increase the utilization of the system.

4.7.2. Resource owner perspective

In Experiment 3, we measured the computational economy related behavior of the system in terms of its supply–demand pattern, resource owner's incentive (earnings) and end-user's QoS constraint satisfaction (average response time and average budget spent) with varying user population distribution profiles. We study the relationship between resource owner's total incentive and end-user's population profile.

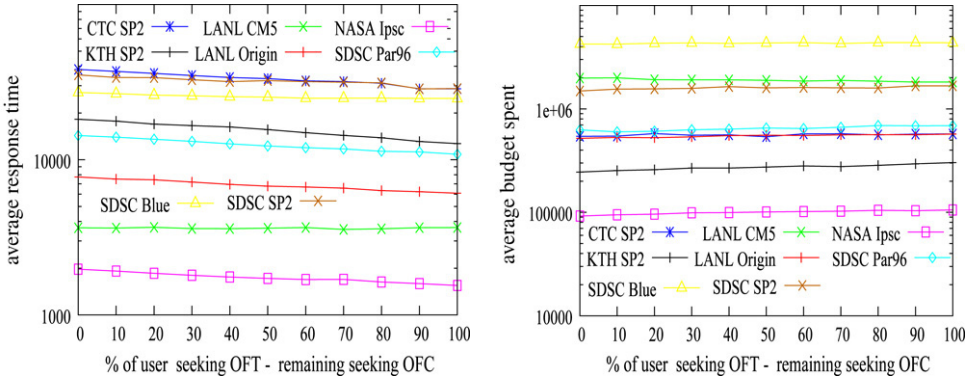
The total incentive earned by different resource owners with varying user population profile can be seen in Fig. 4(a). The result shows as expected that the owners (across all the resources) earned more incentive when users sought OFT (Total Incentive 2.30×10^9 Grid Dollars) as compared to OFC (Total Incentive 2.12×10^9 Grid Dollars). During OFT, we observed that there was a uniform distribution of the jobs across all the resources (refer to Fig. 5(a)) and every resource owner earned some incentive. During OFC, we observed a non-uniform distribution of the jobs in the federation (refer to Fig. 5(a)). We observed that the resources including CTC



(a) Average response time (Sim Units) vs. user population profile.

(b) Average budget spent (Grid Dollars) vs. user population profile.

Fig. 7. Federation user perspective: Excluding rejected jobs.



(a) Average response time (Sim Units) vs. user population profile.

(b) Average budget spent (Grid Dollars) vs. user population profile.

Fig. 8. Federation user perspective: Including rejected jobs.

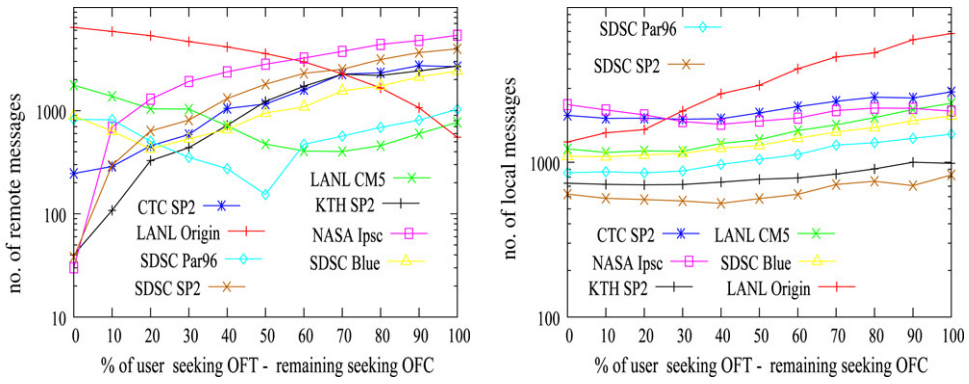
SP2, LANL CM5, LANL Origin, SDSC par96 and SDSC Blue earned significant incentives. This can also be observed in their resource utilization statistics (refer to Fig. 5(a)). However, the faster resources (e.g. KTH SP2, NASA iPSC and SDSC SP2) remained largely underutilized and did not get significant incentives.

Furthermore, the results indicate an imbalance between the resource supply and demand pattern. As the demand was high for the cost-effective resources compared to the time-effective resources, these time-effective resources remained largely underutilized. In this case, the majority of jobs were scheduled on the cost-effective computational resources (LANL CM5, LANL Origin, SDSC Par96 and SDSC Blue). This is the worst case scenario in terms of resource owner's incentive across all the resources in the federation. Although, when the majority of end-users sought OFT (more than 50%), we observed uniform distribution of jobs across resources in the federation. Every resource owner across the federation received significant incentive (refer to Fig. 4(a)) and had improved resource utilization (refer to Fig. 5(a)). These scenarios show balance in the resource supply and demand pattern.

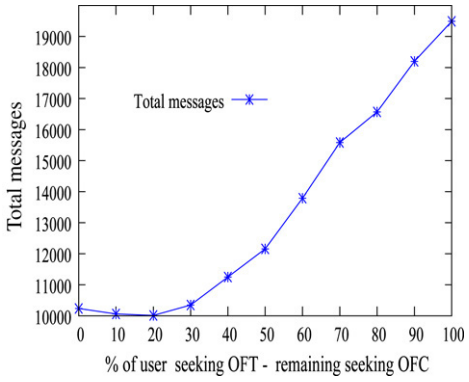
Further, in this case (the majority of users sought OFT (more than 50%)), the average resources in terms of cost/time effectiveness (SDSC Par96 and SDSC Blue) made significant incentive (which can also be seen in their average utilization) as compared to when OFC users constituted the majority population. Probably, this is due to the computational strength of the cost-effective resources (since LANL Origin and LANL CM5 offered 2048 and 1024 nodes, therefore collectively they satisfied the majority of end-users). So, when OFT users formed the majority it resulted in increased inflow of the remote jobs to these average resources. Similar trends can be identified in their respective total remote job service count (refer to Fig. 4(b)). Note that, total remote job service count for cost-effective computational resources (LANL Origin, LANL CM5) decreased considerably as the majority of end-users sought OFT (refer to Fig. 4(b)).

Fig. 6 shows job migration characteristics at various resources with different population profile. We observed that the most cost-efficient resource (LANL Origin) experienced increased job migration rate in the federation as the majority of its users opted for OFT. Conversely, for the most time-efficient resource (NASA iPSC) we observed slight reduction in the job migration rate.

Thus, we conclude that resource supply (number of resource providers) and demand (number of resource consumers and QoS constraint preference) pattern can determine resource owner's overall incentive and his resource usage scenario.



(a) No. of remote messages vs. user population profile. (b) No. of local messages vs. user population profile.



(c) Total messages vs. user population profile.

Fig. 9. Remote-local message complexity.

4.7.3. End users' perspective

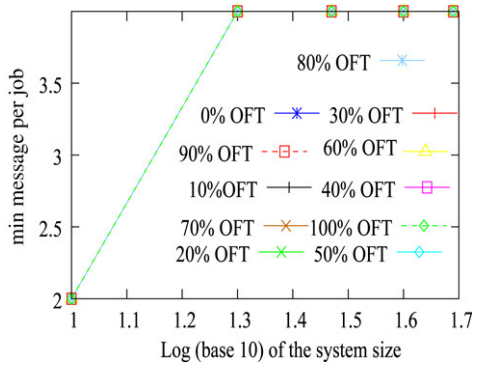
We measured end-users' QoS satisfaction in terms of the average response time and the average budget spent under OFC and OFT. We observed that the end-users experienced better average response times (excluding rejected jobs) when they sought OFT for their jobs as compared to OFC (100% users seek OFC) (scenario-1). At LANL Origin (excluding rejected jobs) the average response time for users was 7.865×10^3 simulation seconds (scenario-1) which reduced to 6.176×10^3 for OFT (100% users seek OFT) (refer to Fig. 7(a)). The end-users spent more budget in the case of OFT as compared with OFC (refer to Fig. 7(b)). This shows that users get more utility for their QoS constraint parameter response time, if they are willing to spend more budget. Overall, the end-users across all the resources in the federation experienced improved response time when the majority constituted OFT population. The end-users belonging to resource LANL CM5 did not have significant change in their response time even with OFT preference. It may be due to their job arrival pattern, that may have inhibited them from being scheduled on the time-efficient resources (though we need to do more investigation including job arrival pattern and service pattern at various resources in order to understand this).

Note that, Fig. 8(a) and (b) includes the expected budget spent and the response time for the rejected jobs assuming that they are executed on the originating resource. Fig. 5(b) depicts the number of jobs rejected across various resources during economy scheduling. During this experiment, we also quantified the average response time and the average budget spent at the fastest (NASA iPSC) and the cheapest resource (LANL Origin) when they are not part of the Grid-Federation (without federation). We observed that the average response time at NASA iPSC was 1.268×10^3 (without federation) simulation seconds as compared to 1.550×10^3 (refer to Fig. 8(a)) simulation seconds during OFT (100% users seek OFT) (as part of federation). Accordingly, at LANL Origin the average budget spent was 4.851×10^5 (without federation) Grid Dollars as compared to 5.189×10^5 (refer to Fig. 8(b)) Grid Dollars during OFC (100% users seek OFC) (as part of the federation). Note that, the plots Fig. 8(a) and (b) do not include the average response time and the budget spent for without federation case.

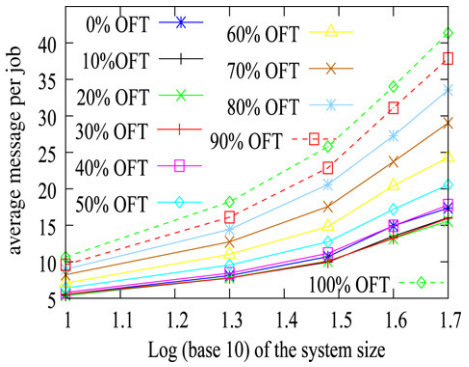
Clearly, this suggests that although federation-based resource sharing leads to better optimization of objective functions for the end-users across all the resources in the federation, sometimes it may be a disadvantage to the users who belong to the most efficient resources (in terms of time or cost).

4.7.4. Remote and local message complexity

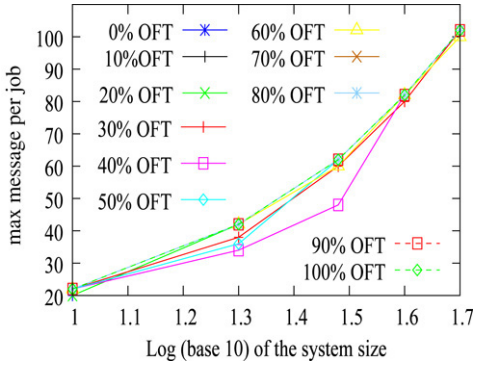
In Experiment 4, we measured the total number of messages sent and received at various GFAs in the federation with varying user population profiles. Fig. 9 shows the plot of the local and remote message count at various GFAs in the federation during economy scheduling. When 100% users seek OFC,



(a) Min messages per job vs. system size.

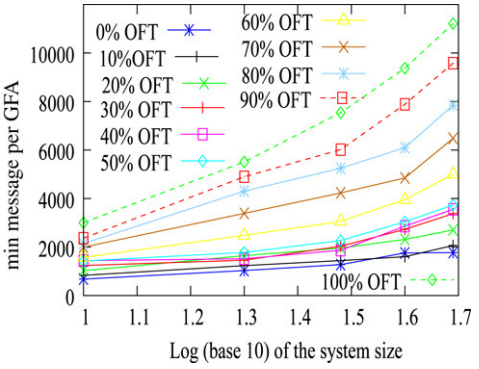


(b) Average messages per job vs. system size.

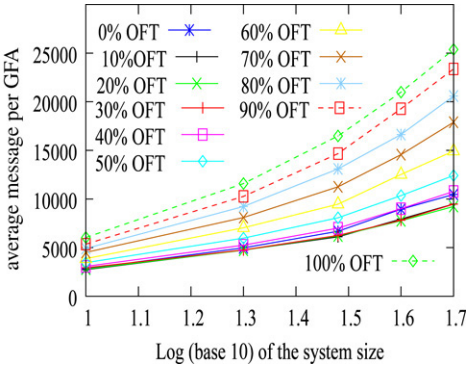


(c) Max messages per job vs. system size.

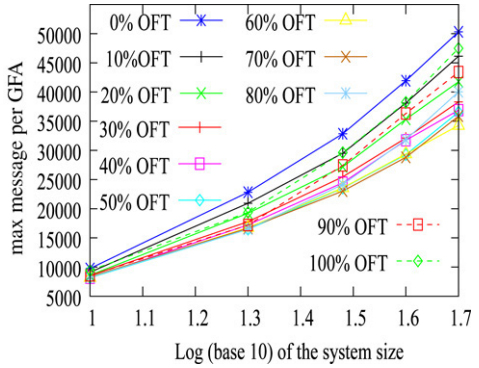
Fig. 10. System's scalability perspective: Message complexity per job with increasing system size.



(a) Min messages per GFA vs. system size.



(b) Average messages per GFA vs. system size.



(c) Max messages per GFA vs. system size.

Fig. 11. System's scalability perspective: Message complexity per GFA with increasing system size.

we observed that resource LANL Origin received maximum remote messages (6.407×10^3 messages) (refer to Fig. 9(a)) followed by LANL CM5 (the second cheapest). LANL Origin offers the least cost, so in this case every GFA in the federation attempted to migrate their jobs to LANL Origin, hence leading to increased inflow of the remote messages. While when 100% users seek OFT, we observed maximum number of remote messages at the resource NASA iPSC (refer to Fig. 9(a)) followed by SDSC SP2 (the second fastest). Since, these resources were time-efficient, therefore all the GFAs attempted to transfer their jobs to them. The total messages involved during this case was 1.948×10^4 as compared to 1.024×10^4 during OFC. This happened because the resources LANL Origin and LANL CM5 had 2048 and 1024 computational nodes and a fewer number of negotiation messages were undertaken between the GFAs for the job scheduling.

Fig. 9(b) shows the total number of local messages undertaken at a resource for scheduling work. The results show, as more users sought OFT, it resulted in increased local message count at cost-effective resources (LANL Origin, LANL CM5). Conversely, faster resources experienced greater remote message count. When 50% sought OFC and 50% sought OFT, we observed uniform distribution of local and remote messages across the federation (refer to Fig. 9(a) and (b)).

To summarize, we observed a linear increase in the total message count with increasing number of the end-users seeking OFT for their jobs (refer to Fig. 9(c)). Hence, this suggests that the resource supply and demand pattern directly determines the total number of messages undertaken for the job scheduling in the computational economy-based Grid system.

Overall, it can be concluded that the population mix of users in which 70% seek OFC and 30% seek OFT seems most suitable from the system and a resource owner perspective. In this case, we observed uniform distribution of jobs, incentives across the resources. Further, this population mix does not lead to excessive message count as compared to other population mix having greater percentage of users seeking OFT.

4.7.5. System's scalability perspective

In Experiment 5, we measured the proposed system's scalability with increasing numbers of resource consumers and resource providers. The first part of this experiment is concerned with measuring the average number of messages required to schedule a job in the federation as the system scales. We observed that at a system size of 10, OFC scheduling required an average 5.55 (refer to Fig. 10(b)) messages as compared to 10.65 for OFT (Fig. 10(b)). As the system scaled to 50 resources, the average message complexity per job increased to 17.38 for OFC as compared to 41.37 during OFT. This suggests that OFC job scheduling required lesser number of messages than OFT job scheduling, though we need to do more work to determine whether this is due to other factors such as budgets/deadlines assigned to jobs. We also measured the average number of (sent/received) messages at a GFA while scaling the system size (refer to Fig. 11). During OFC with 10 resources, a GFA sent/received an average 2.836×10^3 (refer to Fig. 11(b)) messages to undertake scheduling work in the federation as compared to 6.039×10^3 (refer to Fig. 11(b)) messages during OFT. With 40 resources in the federation, the average message count per GFA increased to 8.943×10^3 for OFC as regards 2.099×10^4 messages for OFT.

Figs. 10(b) and 11(b) suggest that the user population including 10%, 20% or 30% OFT seekers involves less number of messages per job/per GFA basis in comparison to 0% OFT seekers. However, further increase in OFT seekers generate more messages per-job/per-GFA basis.

From Figs. 10(b) and 11(b), note that the average message count grows relatively slowly to an exponential growth in the system size. Thus, we can expect that the average message complexity of the system is scalable to a large system size. More analysis is required to understand the message complexity in this case. However, the maximum message count suggests that some parts of the system are not scalable and we need to do more work to avoid these worst cases, e.g. by incorporating more intelligence into the shared federation directory.

Overall, we averaged the budget spent for all the users in the federation during OFC and without federation (independent resources). We observed that during OFC, the average budget spent was 8.874×10^5 Grid Dollars (we included the expected budget spent of rejected jobs on the originating resource) as compared to 9.359×10^5 during without federation. However, at the most popular resource (LANL Origin) the average budget spent for local users during OFC was 5.189×10^5 as compared to 4.851×10^5 during without federation. Similarly, we averaged the response time for all the users in the federation during OFT and without federation. We observed that during OFT, the average response time was 1.171×10^4 simulation units (we included the expected response time of rejected jobs on the originating resource) as compared to 1.207×10^4 during without federation. But at the most popular resource (NASA iPSC) the average response time for local users during OFT was 1.550×10^3 as compared to 1.268×10^3 during without federation. Clearly, this suggests that while some users that are local to the popular resources can experience higher cost or longer delays during the federation- based resource sharing, the overall users' QoS demands across the federation are better met.

5. Conclusion

We proposed a new computational economy-based distributed cluster resource management system called GridFederation. The federation uses agents that maintain and access a shared federation directory of resource information. A cost–time scheduling algorithm was applied to simulate the scheduling of jobs using iterative queries to the federation directory. Our results show that, while the users from popular (fast/cheap) resources have increased competition and therefore a harder time to satisfy their QoS demands, in general the system provides an increased ability to satisfy QoS demands over all users. The result of the QoS-based resource allocation algorithm indicates that the resource supply and demand pattern affects resource provider's overall incentive. Clearly, if all users are seeking either time/cost optimization then the slowest/most

expensive resource owners will not benefit as much. However if there is a mix of users, some seeking time and some seeking cost optimization then all resource providers gain some benefit from the federation. In our future work we will study as to what extent the user profile can change and how pricing polices for resources lead to varied utility of the system. We will also study how the shared federation directory can be dynamically updated with these pricing policies which can lead to coordinated QoS scheduling.

We analyzed how the resource supply and demand pattern affects the system scalability/performance in terms of total message complexity. In general, the cost–time scheduling heuristic does not lead to excessive messages, i.e. to excessive directory accesses and we expect the system to be scalable. However it is clear that popular resources can become bottlenecks in the system and so we intend to research ways to avoid such bottle-necking behavior, principally by using coordination via the shared federation directory. Overall, the proposed Grid-Federation, in conjunction with a scalable, shared, federation directory, is a favourable model for building large scale Grid systems.

Acknowledgments

We thank our group members at the University of Melbourne – Marcos Assuncao, Al-Mukaddim Khan Pathan, Md Mustafizur Rahman – for their constructive comments on this paper. We shall also like to thank anonymous reviewers for their comments that have immensely helped us to improve the quality of work. This work is supported by the Australian Research Council discovery project grant.

References

[1] D. Abramson, R. Buyya, J. Giddy, A computational economy for grid computing and its implementation in the Nimrod-G resource broker, Future Generation Computer Systems 18 (8) (2002) 1061–1074.

[2] B. Alexander, R. Buyya, Gridbank: A grid accounting services architecture for distributed systems sharing and integration, in: Workshop on Internet Computing and E-Commerce, Proceedings of the 17th Annual International Parallel and Distributed Processing Symposium, IPDPS 2003, April 22–26, Nice, France, IEEE Computer Society Press, USA, 2003.

[3] A.O. Allen, Probability, Statistics and Queuing Theory with Computer Science Applications, Academic Press, INC., 1978.

[4] N. Andrade, W. Cirne, F. Brasileiro, P. Roisenberg, OurGrid: An approach to easily assemble grids with equitable resource sharing, in: Proceedings of the 9th Workshop on Job Scheduling Strategies, in: Lecture Notes in Computer Science, 2003.

[5] A. Auyoung, B. Chun, A. Snoeren, A. Vahdat, Resource allocation in federated distributed computing infrastructures, in: OASIS '04: 1st Workshop on Operating System and Architectural Support for the Ondemand IT InfraStructure, Boston, MA, October 2004.

[6] F. Berman, R. Wolski, The apples project: A status report, in: Proceedings of the 8th NEC Research Symposium, Berlin, Germany, 1997.

[7] B. Bode, D. Halstead, R. Kendall, D. Jackson, PBS: The portable batch scheduler and the maui scheduler on linux clusters, in: Proceedings of the 4th Linux Showcase and Conference, Atlanta, GA, USENIX Press, Berkley, CA, 2000.

[8] A. Raza Butt, R. Zhang, Y.C. Hu, A self-organizng flock of condors, in: SC '03: Proceedings of the 2003 ACM/IEEE Conference on Supercomputing, IEEE Computer Society, Washington, DC, USA, 2003.

[9] R. Buyya, D. Abramson, J. Giddy, H. Stockinger, Economic models for resource management and scheduling in grid computing, Concurrency and Computation: Practice and Experience 14 (2002) 13–15.

[10] R. Buyya, 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 14 (13–15) (2002) 1175–1220.

[11] M. Cai, M. Frank, J. Chen, P. Szekely, Maan: A multi-atribute addressable network for grid information services, in: Proceedings of the Fourth IEEE/ACM International Workshop on Grid Computing, 2003, pp. 184–191.

[12] H. Casanova, J. Dongara, Netsolve: A network server solving computational science problem, International Journal of Supercomputing Applications and High Performance Computing 11 (3) (1997) 212–223.

[13] S. Chapin, J. Karpovich, A. Grimshaw, The legion resource management system, in: Proceedings of the 5th Workshop on Job Scheduling Strategies for Parallel Processing, San Juan, Puerto Rico, 16 April, Springer, Berlin, 1999.

[14] J. Chase, L. Grit, D. Irwin, J. Moore, S. Sprenkle, Dynamic virtual clusters in a grid site manager, in: The Twelfth International Symposium on High Performance Distributed Computing, HPDC-12, June, 2003.

[15] J.Q. Cheng, M.P. Wellman, The WALRAS algorithm: A convergent distributed implementation of general equilibrium outcomes, Computational Economics 12 (1) (1998) 1–24.

[16] B. Chun, D. Culler, A decentralized, secure remote execution environment for clusters, in: Proceedings of the 4th Workshop on Communication, Architecture and Applications for Network-Based Parallel Computing, Toulouse, France, 2000.

[17] J. Feigenbaum, S. Shenker, Distributed algorithmic mechanism design: Recent results and future directions, in: DIALM '02: Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, ACM Press, New York, NY, USA, 2002, pp. 1–13.

[18] D.G. Feitelson, D. Tsafrir, Workload sanitation for performance evaluation, in: IEEE Intl. Symp. Performance Analysis of Systems and Software, Springer-Verlag, London, UK, 2006, pp. 221–230.

[19] I. Foster, C. Kesselman, The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers, USA, 1998.

[20] I. Foster, C. Kesselman, S. Tuecke, The anatomy of the grid: Enabling scalable virtual organizations, International Journal of Supercomputer Applications 15 (3) (2001).

[21] J. Frey, T. Tannenbaum, M. Livny, I. Foster, S. Tuecke, Condor-G: A computation management agent for multi-institutional grids, in: 10th IEEE International Symposium on High Performance Distributed Computing, HPDC-10 '01, 2001, IEEE Computer Society, Washington, DC, USA, 2001, pp. 237–246.

[22] W. Gentzsch, Sun grid engine: Towards creating a compute power grid, in: CCGrid, 00:35, 2001.

[23] D. Grosu, T. Chronopoulos, Algorithmic mechanism design for load balancing in distributed systems, IEEE Transactions on Systems Man and Cybernetics Part B (2004) 77–84.

[24] F. Howell, R. McNab, Simjava: A discrete event simulation package for java applications in computer systems modeling, in: First International Conference on Web-Based Modeling and Simulation, San Diego, CA, 1998, SCS Press, San Diego, CA, 1998.

[25] A. Iamnitchi, I. Foster, On fully decentralized resource discovery in grid environments, in: International Workshop on Grid Computing, Denver, CO, 2001.

[26] J. In, P. Avery, R. Cavanaugh, S. Ranka, Policy based scheduling for simple quality of service in grid computing, in: Proceedings of the 18th International Parallel and Distributed Processing Symposium, IPDPS'04, 2004.

[27] K. Lai, B.A. Huberman, L. Fine, Tycoon: A distributed market-based resource allocation system, Technical Report, HP Labs, 2004.

[28] J. Litzkow, M. Livny, M.W. Mukta, Condor — A Hunter of Idle Workstations, IEEE, 1988.

[29] D. Moore, J. Hebeler, Peer-to-Peer: Building Secure, Scalable, and Manageable Networks, McGraw-Hill, Osborne, 2001.

[30] R. Ranjan, A. Harwood, R. Buyya, Grid-Federation: A resource management model for cooperative federation of distributed clusters, Technical Report, GRIDS-TR-2004-10, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, 2004.
[31] A. Rowstron, P. Druschel, Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems, in: Middleware'01: Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms, Heidelberg, Germany, 2001, pp. 329–359.
[32] J.M. Schopf, Ten actions when superscheduling, in: Global Grid Forum, 2001.
[33] H. Shan, L. Oliker, R. Biswas, Job superscheduler architecture and performance in computational grid environments, in: SC '03: Proceedings of the 2003 ACM/IEEE Conference on Supercomputing, IEEE Computer Society, Washington, DC, USA, 2003, p. 44.
[34] J. Sherwani, N. Ali, N. Lotia, Z. Hayat, R. Buyya, Libra: A computational economy-based job scheduling system for clusters, Software: Practice and Experience 34 (6) (2004) 573–590.
[35] S. Smale, Convergent process of price adjustment and global newton methods, American Economic Review 66 (2) (1976) 284–294.
[36] M. Stonebraker, R. Devine, M. Kornacker, W. Litwin, A. Pfeffer, A. Sah, C. Staelin, An economic paradigm for query processing and data migration in maiposa, in: Proceedings of 3rd International Conference on Parallel and Distributed Information Systems, Austin, TX, USA, September 28–30, IEEE CS Press, 1994.
[37] C. Waldspurger, T. Hogg, B. Huberman, J. Kephart, W. Stornetta, Spawn: A distributed computational economy, IEEE Transactions on Software Engineering 18 (2) (1992).
[38] J.B. Weissman, A. Grimshaw, Federated model for scheduling in wide-area systems, in: Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing, HPDC, August 1996, pp. 542–550.
[39] R. Wolski, J.S. Plank, J. Brevik, T. Bryan, G-commerce: Market formulations controlling resource allocation on the computational grid, in: IPDPS '01: Proceedings of the 15th International Parallel & Distributed Processing Symposium, IEEE Computer Society, Washington, DC, USA, 2001, p. 46.
[40] S. Zhou, LSF: Load sharing in large-scale heterogeneous distributed systems, in: Proceedings of the Workshop on Cluster Computing, Tallahassee, FL, 1992.

Rajiv Ranjan is a final-year Ph.D. student in the Peer-to-Peer and Grids laboratory at the University of Melbourne. Prior to joining Ph.D., he completed a bachelor's degree with securing first rank in the Computer Engineering Department (North Gujarat University, Gujarat, India) in 2002. He has worked as a research assistant (honors project) at the Physical Research Laboratory (A Unit of Dept. of Space Govt. of India), Ahmedabad, Gujarat, India. Further he was also a lecturer in the computer engineering department (Gujarat University, Gujarat, India) where he had taught undergraduate computer engineering courses including systems software, parallel computation and advance operating system. His current research interest lies in the algorithmic aspects of resource allocation and resource discovery in decentralised Grid and Peer-to-Peer computing systems. He has served as a reviewer for journals including Future Generation Computer Systems, Journal of Parallel and Distributed Computing, IEEE Internet Computing, and IEEE Transactions on Computer Systems. He has also served as an external reviewer for the conferences including IEEE Peer-to-Peer Computing (P2P'04, P2P'05, P2P'06), IEEE/ACM Grid Computing (Grid'06) and Parallel and Distributed Computing, Applications and Technologies (PDCAT'07).

Dr. Aaron Harwood completed his Ph.D degree at Griffth university on high performamnce interconnection networks in 2002. During that time he worked on several software projects including the development of a VLSI layout package and integrated circuit fabrication virtual laboratory now in use for classroom instruction. He has also worked at Research Institute for Industrial Science and Technology (RIST), South Korea, on computer simulation for a robot traffic light controller system. He then joined the University of Melbourne as a lecturer in the Department of Computer Science and Software Engineering where his research focused on the topological properties and software engineering of peer-to-peer systems for high performance computing. In 2003, he co-founded the Peer-to-Peer Networks and Applications Research Group (www.cs.mu.oz.au/p2p), for which he is now Acting Director. He recently developed one of the first parallel computing platforms for peer-to-peer networks. He is a program committee member for the 6th IEEE/ACM Workshop on Grid Computing.

Dr. Rajkumar Buyya is a Senior Lecturer and the Director of the Grid Computing and Distributed Systems (GRIDS) Laboratory within the Department of Computer Science and Software Engineering at the University of Melbourne, Australia. He received B.E and M.E in Computer Science and Engineering from Mysore and Bangalore Universities in 1992 and 1995 respectively; and Doctor of Philosophy (Ph.D.) in Computer Science and Software Engineering from Monash University, Melbourne, Australia in April 2002. He was awarded Dharma Ratnakara Memorial Trust Gold Medal in 1992 for his academic excellence at the University of Mysore, India. He received Leadership and Service Excellence Awards from the IEEE/ACM International Conference on High Performance Computing in 2000 and 2003. Dr. Buyya has authored/co-authored over 130 publications. He has co-authored three books: Microprocessor x86 Programming, BPB Press, New Delhi, 1995, Mastering C++, Tata McGraw Hill Press, New Delhi, 1997, and Design of PARAS Microkernel. The books on emerging topics that he edited include, High Performance Cluster Computing published by Prentice Hall, USA, 1999; and High Performance Mass Storage and Parallel I/O, IEEE and Wiley Press, USA, 2001. He also edited proceedings of ten international conferences and served as guest editor for major research journals. He is serving as an Associate Editor of Future Generation Computer Systems: The International Journal of Grid Computing: Theory, Methods and Applications, Elsevier Press, The Netherlands. Dr. Buyya served as a speaker in the IEEE Computer Society Chapter Tutorials Program (from 1999–2001) and Founding Co-Chair of the IEEE Task Force on Cluster Computing (TFCC) from 1999–2004, and Interim Co-Chair of the IEEE Technical Committee on Scalable Computing (TCSC) from 2004–Sept 2005, a member of Executive Committee of the IEEE Technical Committee on Parallel Processing (TCPP) from 2004–2005. He is currently serving as Elected Chair of the IEEE Technical Committee on Scalable Computing (TCSC).