A Case for Cooperative and Incentive-Based Coupling of Distributed Clusters

Rajiv Ranjan, Rajkumar Buyya and Aaron Harwood
Department of Computer Science and Software Engineering
University of Melbourne
Victoria, Australia
{rranjan,raj,aharwood}@cs.mu.oz.au

Abstract

Interest in Grid computing has grown significantly over the past five years. Management of distributed cluster resources is a key issue in Grid computing. Central to management of resources is the effectiveness of resource allocation, as it determines the overall utility *of the system. In this paper, we propose a new Grid system that consists of Grid Federation Agents which couple together distributed cluster resources to enable a cooperative environment. The agents use a computational economy methodology, that facilitates QoS scheduling, with a cost-time scheduling heuristic based on a scalable, shared federation directory. 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 [13] 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 and policy heterogeneity, resource discovery, fault tolerance, dynamic resource availability and underlying network conditions.

Existing approaches to resource allocation in a Grid environment are non-coordinated. In a non-coordinated system, application schedulers (e.g. a Resource Brokering System [2]) perform scheduling related activities independent of the other schedulers 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 bottleneck of some valuable resources while leaving others largely underutilized. Furthermore, these brokering systems do not have a co-ordination (or cooperative) mechanism and this exacerbates the load sharing and utilization problems of distributed resources because sub-optimal schedules are likely to occur.

The resources on a Grid (e.g. clusters, supercomputers) are managed by local resource management systems (LRMSes) such as Condor [18] and PBS [5]. These resources can also be loosely coupled to form campus Grids using multi-clustering systems such as SGE [14] that allow sharing of clusters owned by the same organization. In other words, these systems do not allow their combination in the same way that autonomous system support to create an environment for *cooperative federation* of clusters, which we refer as Grid-Federation in rest of the paper.

Furthermore, end-users or their application-level schedulers submit jobs to the LRMS without having the 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 delay and 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) [4]. 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 [10, 23], Condor, NetSolve [9], Punch [17], PBS and SGE provide limited support for QoS driven re-

source 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). They 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, the resource consumer may value various resources differently and may want to negotiate a particular price for using a resource. Users 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.

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 cooperative 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 [19, 15]). In this case the P2P system provides a decentralized database with efficient updates and range query capabilities. Systems like Condor Flock [12] are based on manually configured static resource information and this is a significant disadvantage when building large grid systems. Individual GFAs access the directory information using the interface 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 [20]) 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 [2, 21, 22] for Grid-Federation. In this case resource owners: can clearly define what is shared in the Grid-Federation while maintaining a complete autonomy, can dictate who is given access and get incentives for leasing their resources to federation users. We adopt the market based economic model from [2] for resource allocation in our proposed framework. Other user-centric models are described in [11] and are targeted at clusters. Some of the commonly used economic models [6] in resource allocation includes 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 mainly focus on the commodity market model [24]. In this model every resource has a price, which is based on the demand, supply and value in the Grid-Federation. 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 [20] and QoS constraints.

The key contribution of the paper includes our proposed new distributed resource management model, called Grid-Federation, which provides: (i) decentralization via a shared federation directory that gives site autonomy and scalability; (ii) ability to co-ordinate resource management; (iii) incentives for resources owners to share their resources as part of federation; (iv) 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. In Section 2 we summarize our Grid-Federation and Section 3 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 4.

2 Grid-Federation models

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.

2.1 General Grid-Federation scheduling technique

We define our Grid-Federation (shown in Fig.1) as an architectural framework that enables logical coupling of cluster resources. The Grid-Federation supports policy based transparent sharing of resources and QoS [16] based application scheduling. We also propose a new computational economy metaphor for the Grid-Federation. Computational economy [2, 21, 22] enables the regulation of supply and demand of resources, offers incentive to the resource owners for leasing, and promotes QoS based resource allocation. This framework consists of cluster owners as the resource providers and the end-users as the resource consumers. The 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 distributed information sharing in the Grid-Federation is facilitated through a federation directory. We assume that the the directory information is shared

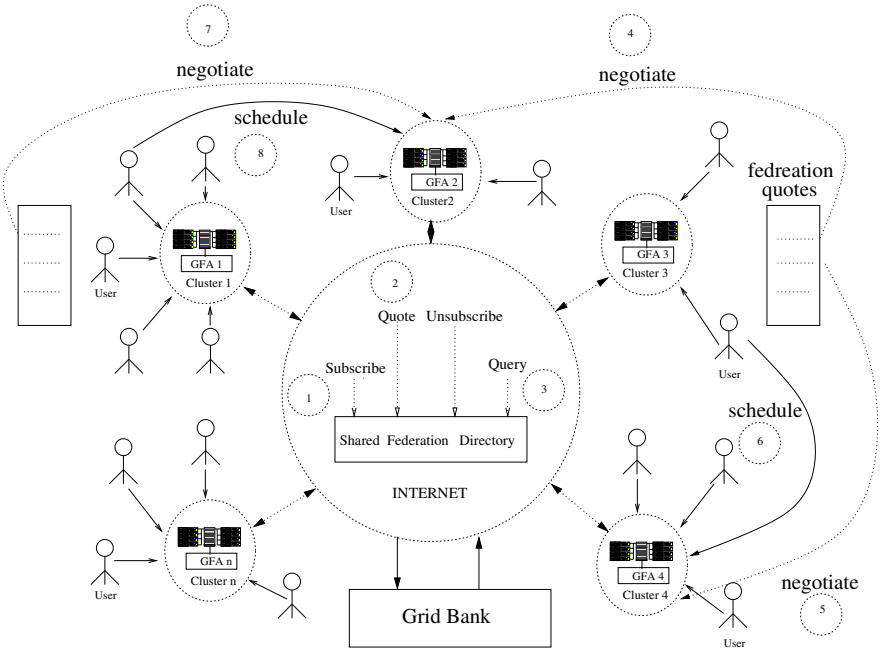


Figure 1. Grid-Federation

using some efficient protocol (e.g. a peer-to-peer protocol [19, 15]). 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 for using that resource c_i, configured by respective cluster owners. Using R_i and c_i, a GFA can determine the cost of executing a job in cluster i and the time taken, assuming that 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 paper, we assume that c_i remains static throughout the simulations. Each GFA can query the federation directory to find the k-th fastest cluster or the k-th cheapest cluster. We assume the query process is optimal, i.e. that it takes $O(\log n)$ messages [8] to query the directory, when there are n GFAs in the system. In this paper, we consider the number of additional messages that are used to satisfy our Grid-Federation scheduling process. We also model Grid Bank [3] as an entity that provides services for credit management in the Grid-Federation.

In Fig.1 a user who is local to GFA 3 is submitting a job. If the user's job QoS can't be satisfied locally then GFA 3 queries the federation directory to obtain the quote of the 1-st fastest or 1-st cheapest cluster. In this case, the federation directory returns the quote advertised by GFA 2. Following this, GFA 3 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 3 queries the federation directory for the 2-nd 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 cluster 4).

Every federation user has to 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 o f 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.

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, $\alpha_{i,j,k}$. We write $J_{i,j,k}$ to represent the i-th job from the j-th user of the k-th resource.

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 com-

munication overhead $\alpha_{i,j,k}$ for a parallel job $J_{i,j,k}$ to be randomly distributed over the processes. In other words, we don't 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

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

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

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

and the associated cost is

$$B(J_{i,j,k}, R_m) = c_m \frac{l_{i,j,k}}{\mu_m \, p_{i,j,k}}. \quad (3)$$

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 time $s_{i,j,k} + d_{i,j,k}$.

2.2 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 r-th fastest GFA; otherwise OFC is required and the query is made for the r-th 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 $B(J_{i,j,k}, R_m) \leq b_{i,j,k}$.

2.3 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. 3),

$$b_{i,j,k} = 2 \times B(J_{i,j,k}, R_k). \quad (4)$$

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

$$d_{i,j,k} = 2 \times D(J_{i,j,k}, R_k). \quad (5)$$

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

3 Experiments and analysis

3.1 Workload and resource methodology

We used trace based simulation to evaluate the effectiveness of the proposed system and the QoS provided by the resource allocation algorithm. The workload trace data was obtained from [1]. The trace contained real time workload of various resources/supercomputers 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 1). The workload trace is a record usage data about collection of parallel jobs that were submitted to various resource facilities. Every job arrives, is allocated

Table 1. Workload and Resource Configuration

Index	Resource / Cluster Name	Trace Date	Processors	MIPS (rating)	Jobs	Quote(Price)	Network Bandwidth (Gb/Sec)
1	CTC SP2	June96-May97	512	850	79,302	4.84	2
2	KTH SP2	Sep96-Aug97	100	900	28,490	5.12	1.6
3	LANL CM5	Oct94-Sep96	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

one or more processors for a period of time, and then leaves the system. Further, every job in the workload has associated arrival time, indicating when it was submitted to the scheduler for consideration. As the experimental trace data [1] 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 GridSim [7] toolkit that allows modeling and simulation of distributed system entities for evaluation of scheduling algorithms. To enable parallel workload simulation with GridSim, we extended existing GridSim's Alloc Policy and Space Shared 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 queue – 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 synthetic QoS specification to each resource including the Quote value (Price that cluster owner charges for service), having varying MIPS rating and underlying network communication bandwidth. The simulation experiments were conducted by utilizing workload trace data over the total period of two days (in simulation units) at all the resources. We consider following resource sharing environment for our experiments:

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

3.2 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 can not be served within requested deadline, then it is rejected otherwise it is accepted. During Experiment 1 and 2, we evaluate the performance of a resource in terms of average resource utilization (amount of real work that resource does over the simulation period excluding the queue processing and idle time), job acceptance rate (total percentage of job accepted) and conversely the job rejection rate (total percentage of job rejected). The result of this experiment can be found in Table 2.

3.3 Experiment 2 – with federation

In this experiment, we analyzed the workload processing statistics of various resources when they are part of the Grid-Federation but do not use 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 3 describes the result of this experiment.

3.4 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, so we assigned them using Eqs. 4,5 to all the jobs across the resources. We performed the experiment under three scenarios:

- all users seek OFC;
- 50% seek OFC 50% seek OFT;

Table 2. Workload Processing Statistics (Without Federation)

Index	Resource / Cluster Name	Average Resource Utilization (%)	Total Job	Total Job Accepted(%)	Total Job 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

- all users seek OFT.

Fig.3 and 4 describes the result of this experiment.

3.5 Experiment 4 – message complexity with respect to jobs

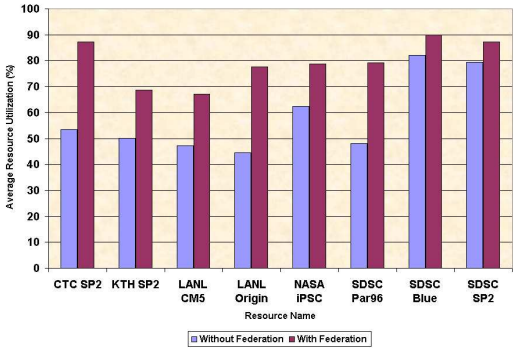
In this experiment, we consider total incoming and outgoing messages at all GFA's. The various message type includes negotiate, reply, job-submission (messages containing actual job) and job-completion (message containing job output). We quantify the number of local messages (sent from a GFA to undertake 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 populations as explained in experiment 3. Fig.5 describes the result of this experiment.

3.6 Experiment 5 – message complexity with with respect to system size

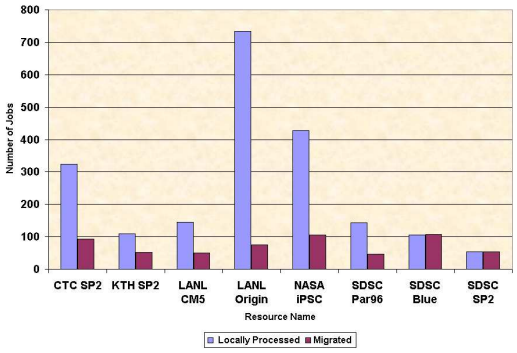
This experiment measures the system's performance in terms of the total message complexity involved as the system size grows from 10 to 50. In this case, we consider the average, max and min number of messages (sent/recv) per Job basis. Note that, in case n messages are undertaken to schedule a job then it involves traversing (if $n > 2$ then $(n-2)/2$, else $n/2$) 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 populations as explained in experiment-3. Fig.6 and 7 describes the result of this experiment. The Java based simulation tool prohibited us from scaling the system further.

3.7 Results and observations

During experiment 1 we observed that 5 out of 8 resources remained underutilized (less than 60%). During experiment 2, we observed that 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



(a) Average resource utilization (%) vs. resource name



(b) No. of jobs vs. resource name

Figure 2. Resource utilization and job migration plot

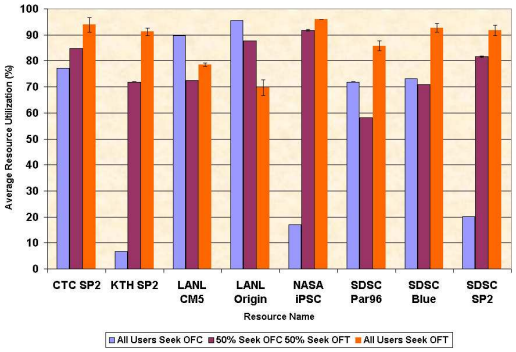
Table 3. Workload Processing Statistics (With Federation)

Index	Resource / Cluster Name	Average Resource Utilization (%)	Total Job	Total Job Accepted(%)	Total Job 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

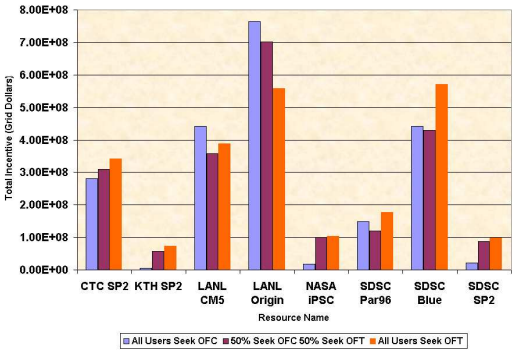
be observed for other resources too (refer to Fig.2(a)). There was an interesting observation regarding migration of the jobs between the resources in the federation (load-sharing). This characteristic was evident at all the resources including CTC SP2, KTH SP2, NASA iPSC etc. At CTC, which had total 417 jobs to schedule, we observed that 324 (refer to Table 3 or Fig.2(b)) of them were executed locally while the remaining 93 jobs migrated and 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 lead to a decrease in the total job rejection rate, this can be observed in case of resource SDSC Blue where the job rejection rate decreased from 42.32% to 1.39%. 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, the job trace shows the potential for resource sharing to increase utilization of the system.

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.3(b). The result shows as expected that the owners (across all the resources) got more incentive when users sought OFT (Total Incentive 2.31×10^9 Grid Dollars) (scenario-3) as compared to OFC (Total Incentive 2.12×10^9 Grid Dollars) (scenario-1). During OFT, we observed that there was a uniform distribution of the jobs across all the resources (refer to Fig.3(a)) and every resource owner got some incentive. While during OFC, we observed a non-uniform distribution of the jobs in the federation (refer to Fig.3(a)). We observed that the resources including CTC SP2, LANL CM5, LANL Origin, SDSC par96 and SDSC Blue earned significant incentives. This can also



(a) Average resource utilization (%) vs. resource name



(b) Total incentive (grid dollars) vs. resource name

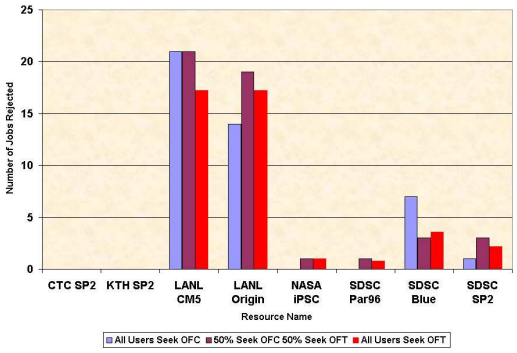
Figure 3. Supply and demand pattern plot

be observed in their resource utilization statistics (refer to Fig.3(a)). However, the faster resources (e.g. KTH SP2, NASA iPSC and SDSC SP2) remained largely underutilized and did not get significant incentives. This is the worst case scenario in terms of the resource owner's incentive across all the resources.

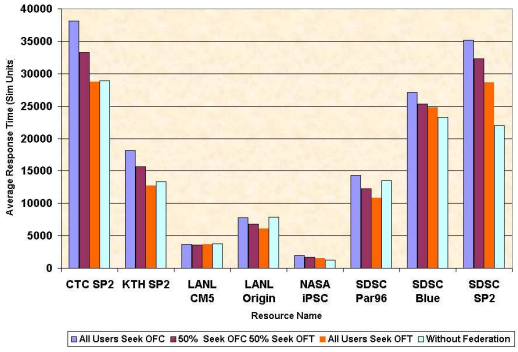
Furthermore, the results indicate an imbalance between the resource supply and demand pattern. As the demand was high for the cost-effective resources compared to the time-effective resources, these time-effective resources remained largely underutilized. In this case, the majority of the jobs were scheduled on the cost-effective computational resources (LANL CM5, LANL Origin, SDSC Par96 and SDSC Blue). Although, with even user population distribution (during scenario-2) all the resource owners across the federation received incentive (Total Incentive 2.16×10^9 Grid Dollars) and had better resource utilization (refer to Fig.3(a)). This scenario shows a balance in the resource supply and demand pattern. Thus, we conclude that resource supply (number of resource providers) and demand (number of resource consumers and QoS constraint preference) pattern can determine the resource owner's overall incentive and his resource usage scenario.

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 received better average response times (excluding rejected jobs) when they sought OFT (scenario-3) for their jobs as compared to OFC (scenario-1). At LANL Origin (excluding rejected jobs) the average response time for the users was 7.865×10^3 simulation seconds (scenario-1) which reduced to 6.201×10^3 for OFT. The end-users spent more budget in the case of OFT as compared OFC (refer to Fig.4(b)). This shows that users get more utility for their QoS constraint parameter response time, if they are ready to spend more budget.

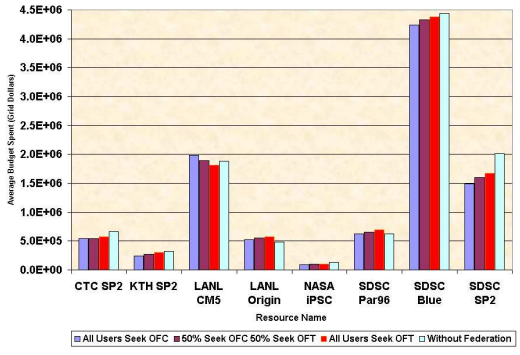
Note that, Fig.4(b) and 4(c) includes the expected budget spent and response time for the rejected jobs assuming they are executed on the originating resource. Fig.4(a) depicts the number of jobs rejected across various resources during economy scheduling. During this experiment, we also consider the 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 1.553×10^3 simulation seconds during OFT (as part of federation) (refer to Fig.4(b)). Accordingly, at LANL Origin the average budget spent was 4.851×10^5 (without federation) Grid Dollars as compared to 5.189×10^5 Grid Dollars during OFC (as part of the federation) (refer to Fig.4(c)). 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



(a) No. of Jobs Rejected vs. resource name

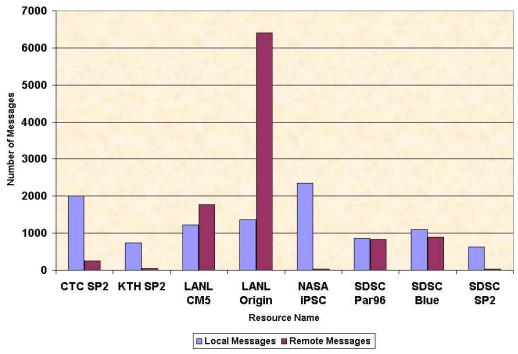


(b) Average response time (simulation units) vs. resource name

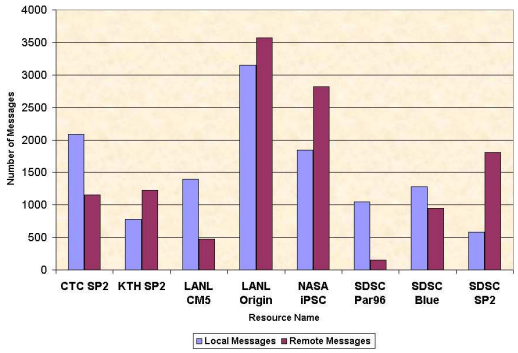


(c) Average budget spent (grid dollars) vs. resource name

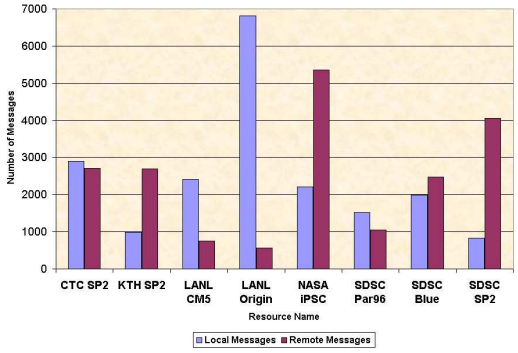
Figure 4. Supply and demand pattern plot



(a) No. of messages (All users seek OFC) vs. resource name



(b) No. of messages (50% seek OFC and 50% seek OFT) vs. resource name



(c) No. of messages (All users seek OFT) vs. resource name

Figure 5. Local and Remote message complexity plot

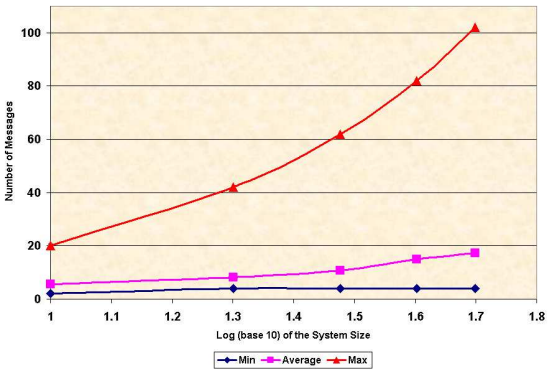
time or cost).

In experiment 4, we measured the total number of messages sent and received at various GFA's in the federation with varying user population profiles. Fig.5 shows the plot of the local and remote message count at various GFAs in the federation during economy scheduling. During (Scenario-1) when all users seek OFC, we observed that resource LANL Origin received maximum remote messages (6.407×10^3 messages) (refer to Fig.5(a)) followed with 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 during (Scenario-3) when all users seek OFT, we observed maximum number of remote messages at the resource NASA iPSC (refer to Fig.5(c)) 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 1.964×10^4 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 GFA's for the job scheduling. During (Scenario-2) when 50% seek OFC and 50% seek OFT, we observed uniform distribution of local and remote messages across the federation (refer to Fig.5(b)). 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.

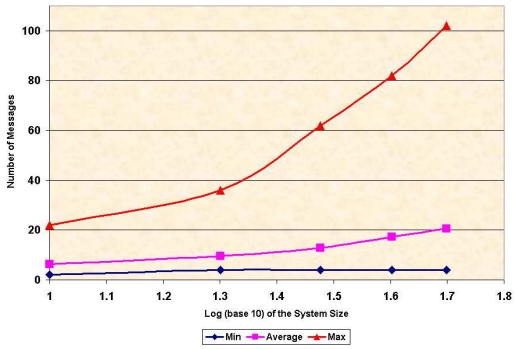
In experiment 5, we measured the proposed system's scalability with increasing numbers of resource consumers and resource providers. The first part of this experiment is concerned with measuring the average number of messages required to schedule a job in the federation as the system scales. We observed that at a system size of 10, OFC scheduling required an average 5.55 (refer to Fig.6(a)) messages as compared to 10.65 for OFT (refer to Fig.6(c)). 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 less 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.

From figure 6, 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 the 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.

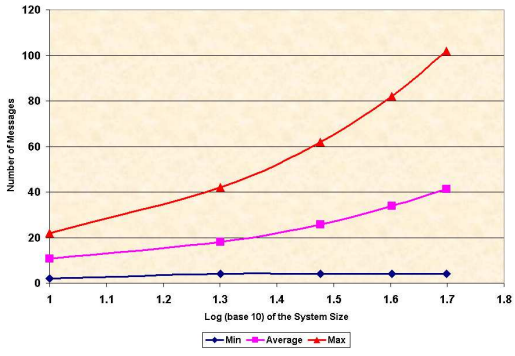
Overall, we averaged the budget spent for all users in the federation during OFC and without federation (in-



(a) No. of messages per job (All users seek OFC) vs. system size



(b) No. of messages per job (50% seek OFC and 50% seek OFT) vs. system size



(c) No. of messages per job (All users seek OFT) vs. system size

Figure 6. Average message complexity per job with increasing system size

dependent resources). We observed that during OFC, 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 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 users in the federation during OFT and without federation. We observed that during OFT, 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 1.553×10^3 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 but the overall users' QoS demands across the federation are better met.

4 Conclusion

We proposed a new computational economy based distributed cluster resource management system called Grid-Federation. 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 therefor 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 or cost optimization then the slowest or 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 to what extent the user profile can change and how pricing polices for resources leads 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 co-ordinated 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.

References

[1] *http://www.cs.huji.ac.il/labs/parallel*.

[2] D. Abramson, R. Buyya, and J. Giddy. A computational economy for grid computing and its implementation in the Nimrod-G resource broker. *Future Generation Computer Systems (FGCS) Journal, Volume 18, Issue 8, Pages: 1061-1074, Elsevier Science, The Netherlands, October*, 2002.

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

[4] A. O. Allen. *Probability, Statistics and Queuing Theory with computer science applications*. Academic Press, INC., 1978.

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

[6] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger. Economic models for resource management and scheduling in grid computing. *Special Issue on Grid computing Environment, The Journal of concurrency and Computation:Practice and Experience (CCPE), Volume 14, Issue 13-15, Wiley Press*, 2002.

[7] R. Buyya and M. Murched. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Journal of Concurrency and Computation: Practice and Experience;14(13-15), Pages:1175-1220*, 2002.

[8] M. Cai, M. Frank, J. Chen, and P. Szekely. Maan: A Multi-atribute addressable network for grid information services. *Proceedings of the Fourth IEEE/ACM International workshop on Grid Computing*, 2003.

[9] H. Casanova and J. Dongara. Netsolve: A network server solving computational science problem. *International Journal of Supercomputing Applications and High Perfomance Computing;11(3); Pages:212-223*, 1997.

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

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

[12] D.H.J. Epema, M. Livny, R. van Dantzig, X. Evers, and J. Pruyne. A worldwide flock of condors: Load sharing among workstation clusters. *Future Generation Computer Systems, Vol. 12*, 1996.

[13] I. Foster and C. Kesselman. The grid: Blueprint for a new computing infrastructure. *Morgan Kaufmann Publishers, USA*, 1998.

[14] W. Gentzsh. Sun grid engine: Towards creating a compute power grid. *Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2002.

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

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

[17] N. Kapadia and J. Fortes. Punch: An architecture for web-enabled wide-area network computing. *Cluster computing: The Journal of Networks, Software Tools and Applications;2(2) Pages:153-164*, 1999.

[18] J. Litzkow, M. Livny, and M. W. Mukta. Condor- a hunter of idle workstations. *IEEE*, 1988.

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

[20] R. Ranjan, A. Harwood, and R. Buyya. Grid federation: An economy based distributed resource management system for large-scale resource coupling. *Technical Report, GRIDS-TR-2004-10, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia*, 2004.

[21] M. Stonebraker, R. Devine, M. Kornacker, W. Litwin, A. Pfeffer, A. Sah, and C. Staelin. An economic paradigm for query processing and data migration in maiposa. *Proceedings of 3rd International Conference on Parallel and Distributed Information Systems, Austin, TX, USA, September 28-30, IEEE CS Press*, 1994.

[22] C. Waldspurger, T. Hogg, B. Huberman, J. Kephart, and W. Stornetta. Spawn: A distributed computational economy. *IEEE Transactions on Software Engineering , Vol. 18, No.2, IEEE CS Press, USA, February*, 1992.

[23] J. B. Weissman and A. Grimshaw. Federated model for scheduling in wide-area systems. *Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing (HPDC), Pages:542-550, August*, 1996.

[24] R. Wolski, J. S. Plank, T. Bryan, and J. Brevik. G-commerce: Market formulations controlling resource allocation on the computational grid. *International Parallel and Distributed Processing Symposium (IPDPS), San Francisco, CA, April*, 2001.