Grid Federation: An Economy Based, Scalable Distributed Resource Management System for Large-Scale Resource Coupling

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

1. Introduction

Interest in Grid [21][22] and Peer-to-Peer (P2P) [5] computing has grown significantly over the past five years. Both are concerned with large scale resource sharing and allows a number of competitive and/or collaborative organizations to share their various resources including hardware, software, data etc. These resources range from desktops to powerful clusters of many processing units. Management of cluster resources is a key issue in grid computing while sharing and management of distributed data is of prime importance to P2P networks. Clusters of computers have emerged as mainstream parallel and distributed platforms for high-performance, high-throughput and high-availability computing. Grid [21] computing extends cluster computing idea to wide-area networks. The Grid consists of cluster resources that are usually topologically apart in 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 [23]. Resource sharing on Grid involves collection of resource providers (cluster owners) and resource consumers (end users) unified together towards harnessing power of distributed computational resources. Such sharing mechanisms can be master-worker based or P2P [32] where providers can be consumers as well, extending between any subset of participants. These resources and their users may even be located in different time zones. There are three key types of cluster arrangement [24], which scale from single systems to supercomputer-class compute farms that utilize thousands of processors:

Cluster Grids are the simplest, consisting of one or more systems working together to provide a single point of access to users on a single project or department.

Campus Grids enable multiple projects or departments within an organization to share computing resources. Organizations can use campus grids to handle a wide variety of tasks, from cyclical business processes to rendering and data mining.

Global Grids are a collection of campus grids that cross organizational boundaries to create a very large virtual systems. Users have access to compute power that far exceeds the resources available within their own organization.

The evolution of resource intensive scientific and commercial applications has led many organizations to own their own clusters. There are various national-level (e.g. CSIRO, APAC), state-level (e.g. VPAC, AC3, SAPAC, TPAC, QPSF) and university-level (e.g. Unimelb, ANU) high performance computing (HPC) platforms. In order to harness the computational power of these cluster resources in a efficient manner, a large scale grid system is imperative. With the advancement in networking technology it is possible to couple various cluster resources to form a logical cooperative environment driven by coordination mechanism. This would lead to a greater pool of resources being utilized for various commercial and scientific purposes.

2. Problem Definition

Existing approach to resource allocation in the Grid environment is non-coordinated in nature. Application schedulers (e.g. Resource Brokering System [4]) view Grid as a large pool of resource to which they hold

an exclusive access. They 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. This enforces overutilization or bottleneck for some resources while leaving others largely underutilized. As these brokering systems do not have a transparent co-ordination mechanism, so they lead to degraded load sharing and utilization of distributed resources.

The resources on the Grid (e.g. clusters, supercomputers) are managed by local resource management systems (LRMS) such as Condor [30] and PBS [9]. These resources can also be loosely coupled to form campus Grids using multi-clustering systems such as SGE [24], LSF [2] that allow sharing of clusters owned by the same organization. This makes the resource pool available for usage very limited and restricts one's ability to access or share external resources. Moreover, these systems do not support the cooperative federation of the autonomous clusters to facilitate transparent sharing and load balancing.

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 in for hours before being actually processed, leading to degraded QoS. To minimize 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 and allocates resources accordingly. To perform these tasks effectively, the schedulers require knowledge of how users value their computations and their QoS requirements, which usually varies with time. The Schedulers also need to provide a feedback signal that prevents the user from submitting unbounded amounts of work.

However, the current system-centric [9][15][20][24][30] approaches to batch scheduling used by the LRMS provide limited support for QoS driven resource sharing. The 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 thus are not good measures of how satisfied the users are with their resource allocations. The system-centric schedulers make decisions that are good for the system as a whole. The users are thus unable to express their valuation of resources and QoS parameters. Further, they do not provide any mechanism for resource owners to define what is shared, who is given the access and the scenario under which sharing occurs [23].

3. Proposed Work

We propose a new model for distributed resource management, in particular federation of clusters. A largescale resource sharing grid system that consists of federation of cluster resources created through peer level coupling, called as *Grid-Federation*. This approach hence enables complete decentralization of control, has better scalability and the system is self-organizable and fault-tolerant. We consider a peer-to-peer network model as a basis for modeling a queuing system that describes salient features/behaviour of grid-federation. The proposed grid system is driven by computational economy methodology for clusters and their federation. Computational economy [10] [37] [38] enables regulation of supply and demand of resources, resource owners get incentive for sharing their resource. Further it promotes user's centric resource allocation. User centric model focus on increasing the user's perceived value based on QoS level indicators and user requirements. In this case the users can express their valuation of resources and QoS constraints. User-centric scheduling yields a better level of system performance coupled with existing system-centric policies. Affect on QoS from the use of economic based scheduling policies in the proposed model are studied. We consider the affect of various parameters such as resource owner policy, user requirements, network delay, bandwidth, congestion on the behaviour of our proposed model. This work includes further investigation in computational economy based resource allocation for different pricing and application models. The proposed work includes supporting transparent load balancing and sharing across clusters in the grid-federation based on user-defined QoS constraints and resource owners sharing policies. The QoS indicators are shown to be effective measure of system utility as system scales with increasing numbers of resource provider and consumers including diversity of the user/owner objective functions. We consider job acceptance rate as a fundamental (QoS) indicator for grid systems and study various factors affecting it including resource owner policies, user constraints and underlying network conditions.

3.1. Organization of the Report

The rest of the report is organized as follows. Section 4 models and illustrates the various components that are part of our grid-federation. In Section 5 we provide experimental results and analysis of our proposed economy model and QoS level indicator. In section 6 we mention some of the related works. In Section 7 we provide concluding remarks and our future vision.

4. Definition

This section builds an abstract model of the entities that are part of our grid-federation. We define the grid-federation model in section 4.1. We present the essential definitions before describing the proposed models, starting with basic entities such as a machine, cluster and a RMS. Then in later part of section we focus on analytical modeling of cluster RMS and grid federation agent (GFA). This is followed by description of the market based quoting process between GFAs. Later we model end user's resource and job descriptions. We end the section with grid-federation economy model. In section 4.4 we present new QoS level indicator for grid system. In section 4.5 we provide a definition of the scheduling algorithms that we consider.

4.1. Grid-Federation

The realm of Grid computing is an extension of the existing scalable distributed computing idea: Internetbased networks of topologically and administratively distributed computing resources. Different resource type includes computers, computational clusters, on-line scientific instruments, storage space, data and various applications. These resource can be utilized by resource consumers in order to solve compute-intensive applications. For managing such complex computing environment traditional methodologies to resource allocation that attempt to enhance system-utilization by optimizing system-centric functions is less efficient. They rely on centralized policies that usually need complete system wide state information to enable application scheduling. They do not focus on the realization of objective functions of the resource providers and the resource consumers simultaneously. Therefore, we propose an economy-based methodology for cooperative management of distributed cluster resources in the Grid environment. This approach will enhance both policy and accountability in resource sharing, that would further lead to optimized resource allocation.

Existing Grid systems including (Legion [15], Condor [30] etc.) offer unrestricted access to the Grid resources. This can sometimes lead to "the tragedy of the commons"–A socioeconomic phenomenon whereby the individually "rational" actions of members of a population have a negative impact on the entire population [17]. These Grid infrastructure lacks both policy and accountability as regards to distributed resource sharing. Currently, there is no standard mechanism that can limit system usage and protect it from free-riders who can abuse the system like in P2P file sharing networks [29]. Other Grid systems such as brokering mechanism access resources independent of other brokers in the system, which can lead to over-utilization of some resources, while under-utilization of others. They do not have any kind of co-ordination [3] mechanism hence are inefficient and non-scalable. The possible solution to this can be set of distributed brokers that co-operate and seamlessly work together having a transparent co-ordination mechanism, which is the notion of our proposed system.

We define our Grid-Federation (shown in Fig.1) as a architectural framework for P2P [25] logical coupling of cluster resources that are under different organizations, administrative and time domains, and that supports policy based [16] transparent sharing of resources and QoS [34][27] based application scheduling. We draw the analogy of Grid-Federation to the electric power Grids [18], which includes a limited number of power suppliers with large investment size (cluster owners). It has large population of electric power consumers purchasing power from these suppliers (federation users) and are connected through various transmission lines (Internet). It provides seamless policy-based (pricing for power/resource consumption) service to is users. This framework aims towards optimizing the user-centric performance of the underlying resources. We also propose a new computational economy metaphor for co-operative federation of clusters. Computational economy [4][37][38] enables the regulation of supply and demand of resources, offers incentive to the resource owners for leasing, and promotes QoS based resource allocation. This new and emerging framework consists of cluster owners as the resource providers and the end-users as the resource consumers. The endusers are also likely to be topologically distributed, having different performance goals, objectives, strategies



Fig. 1. Grid-Federation over P2P Network

and demand patterns. We focus on optimizing resource provider's objective and resource consumer's utility functions through quoting mechanism.

We model a underlying P2P Fig.(1) networking infrastructure for Grid-Federation. To model shared database over P2P [13] network we apply the protocol as proposed in the work (which uses Chord protocol to do resource information sharing). The peer-level logical coupling is facilitated by GFA (Grid Federation Agent) component, which acts as cluster's representative to the federation. It quotes for the jobs to other GFAs with its resource description and pricing policy. A quote consists of a QoS guarantee in terms of resources it has to offer, the price it would charge for those resources evaluated by usage over a fixed period of time. We also model Grid Bank [6] that provides services for accounting in the Grid-Federation.

4.2. Models

4

4.2.1. Terms and Definitions

A *Machine* is a single or multiprocessor system with memory, I/O facilities and an operating system. In this paper we define a *cluster* as a collection of homogeneous machines that are interconnected by a high-speed network like megabyte or gigabyte Ethernet [26]. These machines work as integrated collection of resources. They have a single system image spanning over all the machines. A *resource management system* is a entity which manages a set of resources in the grid-federation. The RMS can optimize any of the system-centric or user-centric performance of underlying resources.

4.2.2. Cluster RMS

In our proposed framework, we assume that every cluster has a generalized RMS, such as a sun grid engine [24] (SGE) or portable batch system [9] (PBS) that manages cluster wide resource allocation and application scheduling. Most of the available RMS packages have centralized organization similar to the master-worker pool model. In the centralized organization, there is only one scheduling controller (master node) which initiates system-wide decisions. We denote the mean arrival rate to a cluster RMS job queue by λ_{C_i} as shown in Fig.(2), where i = 1, 2, ..., n is a unique cluster identifier and n is the number of clusters in the system. Each cluster RMS in the federation has a different mean service rate, μ_{C_i} .

4.2.3. Grid Federation Agent

The grid-federation consists of cluster resources distributed across multiple organizations and administrative domains. 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). It is a two layer resource management

system, managing underlying cluster resources in conjunction with a cluster RMS and enabling policy base resource sharing with its other counterparts in the federation, to enable inter-cluster cooperation across different clusters. A cluster can become a member of the federation by instantiating a GFA component. GFA acts as a resource co-ordinator in the federated space, spanning over all the clusters. These GFAs in the federation inter-operate using an agreed communication mechanism.

The model defines two functional units: (1) *peer manager* and (2) *resource manager*. The peer manager performs tasks like resource discovery and advertisement through well defined primitives. It interacts with a distributed shared database over underlying P2P networking framework (shown in Fig.(1)). 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 main primitives include *subscribe*, *quote*, *query*, *configure* and *unsubscribe*:

subscribe(cluster-id) : subscribe to the grid-federation with cluster-id.

configure(**price**) : configure the pricing model for cluster.

quote(res type,price) : quote for the job in the federation (resource and pricing policy advertisement). **query()** : query the shared database for federation resource information (resource discovery). **unsubscribe(cluster-id)** : unsubscribe from the grid-federation.

The resource manager's main function includes resource allocation and application scheduling. It has specific primitives for communicating with its cluster RMS, local users and remote GFAs. They include:

accept(user-id, job-id) : accepts job from local population of users.

send(user-id, job-id,done) : returns job to local population of users.

send(job-id) : sends job to local cluster RMS.

receive(job-id, done) : receives done job from local cluster RMS.

send(job-id, GFA) : sends job to remote GFA.

receive(job-id, GFA,done) : receives done job from remote GFA.

- accept(job-id, GFA) : receives jobs from remote GFA.
- negotiate(job-id, GFA, deadline) : negotiate with remote GFA specifying deadline constraint for the
 job.

The resource manager deals with *local* jobs and *remote* jobs. Local jobs refer to the jobs submitted by the local population of users. While remote jobs refer to the incoming jobs from remote GFAs.

Fig.(2) shows the job queue model of a cluster. We consider P2P network model in order to analyze the proposed job queueing model of grid-federation. Cluster owners configure their scheduling policy at their GFA, which is then propagated within the federation. GFA attempts to optimize user-centric performance on behalf of its local user population in co-ordination with remote GFAs in the federation.

We denote the mean arrival rate of jobs at a GFA as λ_{G_i} . From Fig.(2):

$$\lambda_{G_i} = \sum_{j=1, j \neq i}^n \lambda_{G_j^{out}} + \lambda_{P_i} + \mu_{C_i},\tag{1}$$

where $\lambda_{G_j^{out}}$ is arrival rate of incoming jobs from remote clusters, $j \neq i$, λ_{P_i} is job arrival rate from local user population and μ_{C_i} is arrival rate of locally serviced jobs.

The local user population job arrival rate is denoted by λ_{P_i} . Depending on the user's specified constraints for a given job, the resource manager component of can execute the job locally or transfer the job to another cluster in the federation, if that cluster can satisfy the user's constraints in a better way. μ_{P_i} denotes rate at which jobs are returned to the local user population. We represent this outgoing job transfer rate by $\lambda_{G_i^{out}}$. This also includes the jobs which were serviced at the cluster. Clearly,

$$\lambda_{C_i} = \lambda_{G_i} - \lambda_{G_i^{out}} - \mu_{P_i},\tag{2}$$

where $\lambda_{G_i^{out}}$ is job transfer rate to other clusters.

In general, μ_{C_i} and μ_{G_i} depend on the cluster owner's scheduling policy, hardware and software configurations and network performance. μ_{P_i} is the rate at which done or rejected jobs are returned to the local user population. We use a Poisson arrival rate for λ_{P_i} (local user population) which drives the model response.



Fig. 2. Cluster i of Grid-Federation

We model the job arrival rate at various clusters in federation as a Poisson process and has the distribution of Poisson random variable. The rate λ_{G_i} denotes the mean or average job arrival rate at cluster *i* of federation. At cluster *i* for a time interval [0, t], the probability of n arrivals in t units of time is given by

$$P_n(t) = \frac{\left(\lambda_{G_i} t\right)^n}{n!} e^{-\lambda_{G_i} t}$$
(3)

Federation consists of n clusters having mean job arrival rate λ_{G_i} where i = 1, 2, ..., n. That is n different Poisson processes with distributions for the arrival rate $\frac{(\lambda_{G_i}t)^n}{n!}e^{-\lambda_{G_i}t}$ where i = 1, 2, ..., n. Merging property of Poisson process states that if we merge n Poisson processes into one single process, then the result is a single Poisson process. Merging of above stated Poisson processes will result into a single Poisson process having mean λ_{G_total}

$$\lambda_{G^{total}} = \lambda_{G_1} + \lambda_{G_2} + \dots + \lambda_{G_n} \tag{4}$$

The Inter arrival times of Poisson process has an exponential distribution with mean rate λ_{G_i} . For instance let us pick an arbitrary starting point t_0 in time and T_1 be the time until the next arrival at some cluster *i*. This gives

$$P(T_1 > t) = P_0(t) = e^{-\lambda_{G_i} t}$$
(5)

Thus the cumulative distribution function (cdf) of T_1 is given by

$$F_{T1}(t) = P(T_1 \le t) = 1 - e^{-\lambda_{G_i} t}$$
(6)

And the probability distribution function (pdf) of T_1 is

$$f_{T1}(t) = \lambda_{G_i} e^{-\lambda_{G_i} t} \tag{7}$$

Therefore, T_1 has an exponential distribution with mean rate λ_{G_i} .

If we merge *n* Poisson processes with distributions for the inter arrival times $1 - e^{-\lambda_{G_i}t}$ where i=1, 2, ..., n into one single process, then the result is a Poisson process for which the inter arrival times have the distribution $1 - e^{-\lambda_{G_i}t}$ with mean

$$\lambda_{G^{total}} = \lambda_{G_1} + \lambda_{G_2} + \dots + \lambda_{G_n} \tag{8}$$

4.2.4. User's Job Specification

The user's job specification consists of user's resource requirement and preference for that particular job. A job is described by a directed acyclic or cyclic task graph.



Fig. 3. Directed Acyclic Task Graph

A job is a set of tasks whereas task is any independent program or executable.

$$J_i = \{T_1, T_2, T_3, T_4, \dots, T_n\}$$
(9)

Where n is the number of tasks in the job set J_i .

if n=1 then the job is said to be independent, else the job consists of set of dependent tasks described by the task graph.

The task may be composed of parallel application like MPI or PVM, which can lead to two-way communication during their execution as depicted in Fig.(3) between task T_2 and T_3 . These tasks may execute on the same cluster or on different clusters. They are represented by directed cyclic graphs. PSfrag replacements



Fig. 4. Directed cyclic Task Graph

$$G = (V, E) \tag{10}$$

$$J_i = V = \{T_1, T_2, T_3, T_4, \dots, T_n\}$$
(11)



 b_{ij} is the total data passed between the task i to j during their execution.

4.2.5. Grid-Federation Resource Description

Grid-Federation Resource Description is a set R_G which contains resource description of various clusters in the federation.

$$R_G = \{R_{C_1}, R_{C_2}, \dots, R_{C_n}\}$$
(12)

7

⁸ Where R_{C_i} is the cluster resource description set, *i* varies from 1 to *n* depending on the number of clusters in the grid-federation. Each cluster in the federation has its own Resource set R_{C_i} which contains the definition of all resource owned by the cluster and ready to be offered.

$$R_{C_i} \in C_i \times O_i \times M_i \times S_i \times L_i \times N_i \tag{13}$$

 $R_{C_i} \in \{ \langle c, o, m, s, l \rangle | (c \in C_i) \land (o \in O_i) \land (m \in M_i) \land (s \in S_i) \land (l \in L_i) \land (n \in N_i) \}$ (14)

 C_i is the set describing the type of Central Processing Unit available on cluster. $c \in C_i$ is particular cpu type e.g. c = i386 or Alpha.

 O_i is the set describing operating system type available on the cluster. $o \in O_i$ is particular os type e.g. o = Solaris or Linux.

 M_i is the set describing the amount of physical memory available on the cluster. $m \in M_i$ is RAM size e.g. m= 256 MB .

 S_i is the set describing secondary storage space available on the cluster. $s \in S_i$ is available secondary storage size e.g. s = 1 GB.

 L_i is the set describing the additional library offered by cluster. $l \in L_i$ is particular additional library e.g. l = MPI or PVM.

 N_i is the set describing number of nodes in the cluster. $n \in N_i$ is nodes at particular cluster e.g. n=1 or 4.

4.2.6. User's Resource Description

Users specify its Resource requirement as

$$R_U = \{C_U, O_U, M_U, S_U, L_U\}$$
(15)

4.2.7. Economy Models in Grid-Federation

Existing work in resource management and application scheduling in Grid computing is driven by conventional metaphor where a scheduling component takes decision regarding the site where application will be executed based on some system-centric parameters (Legion [15], Condor [30], Apples [8], NetSolve [14], Punch [28]). They treat all resources with the same scale, as if they 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 different objective function. Similarly the resource consumer may value various resources differently depending on its QoS based utility functions, may want to negotiate a particular price for using a resource based on demand, availability and its budget. To overcome these shortcomings, we propose an economicsbased resource allocation, in this case the scheduling mechanism is driven by resource provider's sharing policy, objective functions and resource consumer's QoS based utility functions. Pricing is primarily based on the demand by the resource consumers and resource availability pattern, in a economic market based resource allocation model.

Some of the commonly used economic model [11] 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 [39]. In this model every resource has a price, which is based on the demand, supply and value in the Grid-Federation. The cost model for the particular cluster depends on the resources it provides to the federation user and is valued accordingly. The initial price of the resources are configured by their owners, it varies between the clusters depending on the hardware configuration, software availability and user's percieveness of QoS.

The relative worth of resources are determined by their comparative supply and demand pattern. If a resource has less demand, then its owner quotes with lower price as compared to previous quote in order to attract more users. Every federation user has to express how much he is willing to pay (*budget*) and expected response time (*deadline*) for his job. User's valuation of resources for his job is directly governed by the job specification and QoS requirements.

Quality is the totality of features of a service that influences its ability to satisfy the given needs. Quality of service evaluations are considered to be driven by a comparison of consumer expectations with their perceptions of the actual quality received. QoS is a guaranteed level of performance delivered to the customer, which is part of service level agreement (SLA) between the service providers and the end-users. The QoS can be characterized by several basic performances criteria including availability, performance, response time and throughput. Service providers may guarantee a particular level of the QoS as defined in the SLA. In our proposed framework the SLA is part of quoting process, in which the cluster owners are committed towards providing the services they define in their subsequent quotes. The focus of user-centric resource allocation is towards maximizing the end-users satisfaction in terms of QoS constraints. Our Grid-Federation economy model defines the cluster owners, $C_{Gowner} = \{c_1^{owner}, c_2^{owner}, ..., c_n^{owner}\}$ that owns resources $R_G = \{R_{c_1}, R_{c_2}, ..., R_{c_n}\}$. Every cluster in the federation has its own Resource set R_{c_i} which contains the definition of all resource owned by the cluster and ready to be offered. R_{c_i} includes information about the CPU architecture, number of processors, RAM size, Secondary storage size and Operating system type. Every resource in federation has a price, which we represent by $P_{Gcost} = \left\{c_1^{price}, c_2^{price}, ..., c_n^{price}\right\}$. The resource owner c_i^{owner} charges c_i^{price} per unit time or price per unit of Million Instructions (MI) executed e.g. per 1000 MI. There is mapping function from set of federation resources (R_G) to cluster price model (P_{Gcost}) .

$$\Pi: R_G \to P_{Gcost} \tag{16}$$

Let $U_G = \{c_1^{user}, c_2^{user}, ..., c_n^{user}\}$ contains the federation users belonging to various clusters. c_i^{user} represents the users belonging to cluster *i*. Every cluster owner c_i^{owner} requires jobs J_u to use its resource power. A user owns a job $J_i \in J_u$. Every federation user u_i is modeled as having a resource allocation utility function $Q_{oS}(Constraint)$ for each job which indicates QoS value delivered to the user as a function of specified QoS constraints (deadline and budget). Each job J_i consumes some power of particular type of cluster resource R_{c_i} .

For every job J_i , federation user u_i determines a *budget*, which he is ready to spend in order to get his job done. This is a mere user's assumption which can be feasible or unfeasible. If this assumption is unfeasible then it is quite likely that user's job would get rejected from the federation, in that case the user may have to increase the budget constraint. In addition to budget, user may also give his preference about the response time it expects from the system *(deadline)*. When users submit their jobs to the GFA, they express maximum value of both budget and deadline constraints with one of the two optimization strategy that should be adopted during scheduling.

Every federation user $u_i \in c_i^{user}$ can express the optimization strategy he intends for his job J_i . We propose two optimization strategies that a user can opt for. Starting with the *Time Optimization* [4] strategy, where the focus is on getting the work done as fast as possible. In this case the users specify the maximum budget (c_{budget}) and the deadline $(t_{deadline})$ for their job. In this optimization strategy the user might get his job done within the deadline limit but he may have to invest maximum budget. This signifies as the user invests more budget, it is likely that he will get better response time from the system.

Sometimes the federation user would like to make use of both of the above strategies but without really maximizing or minimizing either of the time or cost constraints. This is called *Cost-Time Optimization*[4] strategy. In this strategy the user spends a fair amount of the allocated budget for the job, while getting a more acceptable response time from the system as compared to cost optimization.

$$Response - Time \propto 1/Budget \tag{17}$$

The federation user can also specify *Cost Optimization* [4] strategy for his job, in this case focus is on getting the work done in minimum possible cost, but within the time constraint. This strategy will get the user's job done in minimum possible cost while maximizing the response time within the deadline limit.

4.2.8. Quoting Mechanism between GFAs

This framework aims towards P2P coupling of various clusters thus overcoming the burden of central management and thereby giving autonomous control to individual clusters about their functioning. Each of these clusters are driven by different pricing policies.

In Fig.(5), cluster A in the federation does quote broadcast to all other clusters in the federation through P2P shared database. A user who is local to cluster A is making a request while the other clusters are

broadcasting their quote. A typical quote consists of resource description R_{c_i} and c_i^{price} (price to be paid for using the specified cluster resource), configured by cluster owner. After analyzing all the quotes, cluster A decides whether the request should be serviced locally or forwarded to another cluster. In this way cluster A has the information about all other cluster's service policies.

If the user request can not be served locally then cluster A evaluates all quotes against the user's required QoS. After this cluster A sends negotiate message (Enquire about QoS guarantee in terms of response time) to the matching (In terms of the resource type and the service price) clusters (cluster A sending negotiate message to cluster E and D) one by one until it finds the cluster on which it can schedule the job (job finally scheduled on cluster E).

4.3. Supporting Technology : Peer-to-Peer

The concept of P2P systems is a new revolution to the domain of Internet computing. A P2P [7] system is a self-organizing distributed system where self-interested peers communicate among themselves to share resources such as storage, data or CPU time. Moore and Hebeler, in the book [32], have defined P2P as paradigm that supports the exchange of information and services directly between producers and consumers in order to achieve purposeful results. The primary motivation of using P2P technology for our grid-federation is scalability and fault-tolerance. The Internet computing environment is composed of thousands of cluster resources, traditional monolithic approach of coupling them is non-scalable and is liable to one-point failure.

Our model of grid federation over underlying P2P network is shown in Fig.(1). The federation consists of n GFAs interconnected through the P2P [36] network. To model shared database over P2P network we apply the protocol [13] as proposed in the work (which uses Chord protocol to do resource information sharing). The network is logically fully connected and accessible through distributed shared federation directory, and that every GFA can communicate with every other GFA. This shared database is distributed shared memory (DSM) over P2P network and each GFA has its own local copy of database. The grid peer component of GFA interacts with this shared database using defined primitives such as subscribe, query, post and unsubscribe, is responsible for consistency, synchronization, fault tolerance, coherence, and persistence of the shared space.





Fig. 5. Quoting Mechanism in Grid Federation

4.4. Quality of Service Indicator for Grid Systems

To date, the factors that influence QoS, such as cluster owner policy and user constraints have not well studied in the literature. We define acceptance rate as a QoS indicator for Grid Systems and show how the cluster owner policies, resource availability, various economy models and user constraints affect the QoS. If a submitted job can not be completed within its given constraints then it is rejected, otherwise accepted. Acceptance rate is the percentage of all jobs that are accepted. We consider the acceptance rate of our proposed grid-federation.

4.5. QoS Constraint Driven Resource Allocation Heuristic

We propose a deadline and budget constrained(DBC) grid federation scheduling heuristic, called the costtime optimization scheduling. A detailed algorithm for scheduling jobs to cluster resources in the federation, that optimizes cost, time and cost-time follows next. The algorithm aims towards optimizing user-centric performance of underlying cluster resources. The federation user can specify any one of the optimization strategy for their job:

(Our algorithm is an extension of basic Nimrod-G [4] algorithm)

- (1) Optimize for time: The focus of this strategy is to give minimum possible response time to the federation user, but within the budget limit.
- (2) Optimize for cost: This strategy produces results by deadline, but reduces cost within a budget limit.
- (3) Optimize for cost-time: This strategy optimizes both cost and time parameter. In this case the federation user spends a fair amount of the alloted budget for the job while getting a more acceptable response time from the system.

All the scheduling related activities are performed by peer and resource manager component of the GFA. We will explain the scheduling algorithm in the context of these components.

Algorithm

(1) Peer Manager.

- (a) Subscribe: Register to the federation with unique cluster id.
- (b) Quote: Advertise the cluster owner's quote(res type, price) (Resource Advertisement).
- (c) Query: Query the distributed shared database, obtain quotes of other clusters in the federation(Resource Discovery).
- (d) Unsubscribe: Cancel or suspend the membership of cluster from the federation.
- (2) Resource Manager.
 - (a) Analyze Quotes: Identify the resource type, characteristics, configuration, capability and the usage cost per unit time by analyzing the quotes advertised by various clusters in the federation. Store these statistics for future job scheduling in *Federation – Resource – List*.
 - (b) Accept, Analyze and Schedule Local Jobs: Accept local jobs and store in Jobs Wait List. Repeat the following steps for each waiting job Job_i .
 - i. Identify the list of clusters in the federation matching the job's resource requirement from Federation-Resource-List.
 - ii. For each such matching cluster calculate the budget required to execute job on that cluster. If the user of the job supplies Deadline and Budget for the job, then determine the *absolute deadline* and *budget* based on the matching cluster's resource processing capability and pricing policy. Store this in Job-Match-List. Repeat this step for all matching clusters in *Federation Resource List*.

iii. Now determine the optimization requested by the user of the job and dispatch the job.

A. For cost optimization, sort the $Job_i - Match - List$ by increasing order of cost. Then select the first cluster in the list and negotiate whether it can complete the job within the user specified deadline. If yes dispatch the job, remove the job from Jobs - Wait - List and add to Jobs - Submit - List. If no then repeat the same for next cluster in the $Job_i - Match - List$. If at last none of the cluster can complete within specified deadline then add the job to Reject - Jobs - List, remove from Jobs - Wait - List. Return the job to the user.

- B. For time optimization, sort the $Job_i Match List$ by increasing order of absolute deadline. Then select the first cluster in the list and negotiate whether it can complete the job within the user specified budget. If yes dispatch the job, remove the job from Jobs - Wait - List and add to Jobs - Submit - List. If no then repeat the same for next cluster in the $Job_i - Match - List$. If at last none of the cluster can complete within specified budget then add the job to Reject - Jobs - List, remove from Jobs - Wait - List. Return the job to the user.
 - C. For cost-time optimization, determine the cost-time factor ct_i (by multiplying absolute deadline and absolute budget) for each cluster in the user's match list. Now sort the the $Job_i - Match - List$ by increasing order of of ct_i . Then select the first cluster in the list and negotiate whether it can complete the job within the user specified deadline. If yes dispatch the job, remove the job from Jobs - Wait - List and add to Jobs - Submit - List. If no then repeat the same for next cluster in the $Job_i - Match - List$. If at last none of the cluster can complete within specified budget and deadline then add the job to Reject - Jobs - List, remove from Jobs - Wait - List. Return the done job to the user.
- (c) Accept and Schedule Remote Jobs: For each incoming job
 - i. Accept the incoming job, add to Remote Job Wait List.
 - ii. Transfer the job to local cluster RMS for execution, add to Remote Job Submit List. Remove from Remote – Job – Wait – List.
 - iii. On arrival of completed job from cluster RMS, add job to Remote Job Done List, transfer the job to its originating cluster. Remove the job from Remote Job Submit List.
- (d) Receive finished jobs: For each incoming completed job
 - i. Add the job to Jobs Done List. Remove the job from Jobs Submit List.
 - ii. Return the completed job to the user.
 - iii. If the job did not complete successfully, then add it to Jobs Wait List.

4.5.1. Heuristic Analysis

Assignment of job J_i to the resources in the federation can be formally described by the function

$$\Delta: J_U \longrightarrow R_G \tag{18}$$

from the set of jobs J_U to the set of federation resource R_G .

At any time t given m jobs $J_1, J_2, ..., J_m$ and p clusters resources $R_{c_1}, R_{c_2}, ..., R_{c_p}$ that matches jobs resource and QoS requirements, it is possible to assign them in p^m ways. Each job J_i has c_{budget} and $t_{deadline}$ associated with it. The problem is to find an a resource, which minimizes both c_{budget} and $t_{deadline}$ in accordance with the optimization strategy sought by the owner of the job J_i . Further the assignment strategy should lead to efficient utilization of federation resources and minimize the job starvation rate.

Resource allocation for job J_i can be optimized of any of the two user specified QoS constraints. We define R_{cost} as a function which determines the processing cost of resource R_{c_i} (service price), R_{power} as a function which determines the processing power of resources R_{c_i} and R_{factor} as a function which determines the processing power of resources R_{c_i} and R_{factor} as a function which determines the processing power of resources R_{c_i} .

$$R_{cost}: R_{c_i} \longrightarrow Q \tag{19}$$

$$R_{power}: R_{c_i} \longrightarrow Q \tag{20}$$

$$R_{factor}: R_{c_i} \longrightarrow Q \tag{21}$$

If user seeks cost optimization for his job then, allocate resource R_{c_k} , k < p, such that,

$$R_{cost}(R_{c_k}) = min(R_{cost}(R_{c_i})) \quad i = 1...p$$

$$\tag{22}$$

If user seeks time optimization for his job then, allocate resource R_{c_k} , k < p, such that,

$$R_{power}(R_{c_k}) = max(R_{power}(R_{c_i})) \quad i = 1...p$$

$$\tag{23}$$

If user seeks cost-time optimization for his job then, allocate resource R_{c_k} , k < p, such that,

$$R_{factor}(R_{c_k}) = min(R_{cost}(R_{c_i}) * R_{power}(R_{c_i})) \quad i = 1...p$$

$$\tag{24}$$

12

Index	Resource	Trace Date	Nodes	MIPS	Jobs	Quote(Price)
	/ Cluster			(rat-		
	Name			ing)		
1	CTC SP2	June96-May97	512	850	79,302	5.0
2	KTH SP2	Sep96-Aug97	100	900	28,490	5.2
3	LANL	Oct94-Sep96	1024	700	$201,\!387$	3.6
	CM5					
4	LANL	Nov99-Apr2000	2048	630	$121,\!989$	3.5
	Origin					
5	NASA	Oct93-Dec93	128	930	42,264	5.3
	iPSC					
6	SDSC	Dec95-Dec96	416	710	38,719	3.6
	Par96					
7	SDSC	Apr2000-Jan2003	1152	730	250,440	3.7
	Blue					
8	SDSC	Apr98-Apr2000	128	920	$73,\!496$	4.5
	SP2					

Table 1. Workload and Resource Configuration

The following holds true for all optimization strategy. Let the start time of J_i is s_i , (we assume that the s_i 's are integer, and that min $\{s_i\}=0$)

Every job J_i has deadline $t_{deadline}$ and budget c_{budget} so,

$$s_i + \tau_i \le t_{deadline} \tag{25}$$

$$\tau_i = Total \ CPU \ Time \ required \ by \ the \ job \tag{26}$$

and,

$$J_i^{p-cost} = R_{cost}(R_{c_i}) \cdot \tau_i \le c_{budget}$$

$$\tag{27}$$

$$J_i^{p-cost} = c_i^{price} \,.\, \tau_i \le c_{budget} \tag{28}$$

 J_i^{p-cost} denotes processing cost of job J_i on the resource R_{c_i}

5. Experiment and Analysis

We used trace based simulation to evaluate the effectiveness of the proposed system and the QoS provided by the resource allocation algorithm. The simulator was implemented using GridSim [12] toolkit that allows modeling and simulation of distributed system entities for evaluation of scheduling algorithms. Our simulation environment models the following basic entities in addition to existing entities in GridSim:

- (1) Local user population, which basically models the local user population.
- (2) GFA, generalized RMS system that we model for Grid-Federation.
- (3) GFA queue, placeholder for incoming jobs from local user population and the federation.
- (4) GFA shared federation directory over Peer-to-Peer network, for distributed information management.

5.1. Workload and Resource Modeling

We based our experiments on real time workload trace data obtained from [1] various resources/supercomputers (See Table-I). The trace data was composed of parallel applications. To enable parallel workload simulation with GridSim, we extended existing GridResource, Alloc Policy and Space Shared entities. 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) and having varying MIPS rating. The simulation experiments were conducted by utilizing workload trace data over the total period of two days (in simulation units) at all the resources. In experiment 1 and 2 we consider, \mathbb{H}^4 the user request can not be served when requested, then it is rejected otherwise it is accepted. During experiment-1 and experiment-2 we measure the following

- (1) Average resource utilization (Amount of real work that resource does over the simulation period excluding the queue processing and idle time).
- (2) Job acceptance rate (total percentage of job accepted).
- (3) Job rejection rate (total percentage of job rejected).
- (4) Number of jobs locally processed.
- (5) Number of local jobs migrated to federation.
- (6) Number of remote jobs processed.

5.2. Experiment-1 (Independent Resource)

In this experiment the resources were modeled as an independent entity (without federation). All the workload submitted to the resources was processed locally. We evaluate the performance of a resource in terms of average resource utilization, job acceptance rate and job rejection rate. The result of this experiment can be found in (refer to Table-2). We observed that about half of the resources including CTC, KTH SP2, LANL Origin, NASA iPSC, and SDSC Par96 were utilized less than 50%.



Fig. 6. Average Resource Utilization (%) Vs. Resource Name

5.3. Experiment-2 (With Federation)

In this experiment we analyzed the workload processing statistics of various resources when they are part of the Grid-Federation, in this case the workload assigned to the resource can be processed locally or may



Fig. 7. No. of Jobs Vs. Resource Name

be migrated to any other resource in the federation depending on the availability pattern. Table-3 describes the result of this experiment.

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 increased from mere 36.71% to 85.85%. Same trends can be observed in case of other resources too (refer to Fig.(6)). 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, KTH SP2, NASA iPSC etc. At CTC, which had total 417 jobs to schedule, we observed that 383 (refer to Table-3) of them were executed locally while the remaining 34 jobs migrated and executed at some remote resource in the federation. Also, this resource executed 80 remote jobs, which came from other resources in the federation.

The federation based load-sharing also led to decrease in the total job rejection rate, this can be observed in case of resource LANL CM5 whose job rejection rate decreased from 18.83% to 0.093%. Thus, we conclude that the federation based resource allocation promotes transparent load-sharing between various participant resources, which further helps in enhancing their overall resource utilization and the job acceptance rate.

5.4. Experiment-3 (With Federation and Economy)

In this experiment, we study the computational economy metaphor in the Grid-Federation. We assigned QoS parameters (budget and deadline) to all the jobs across the resources. We performed the experiment under three scenarios having different user population profile.

- (1) All users seek cost-optimization.
- (2) Even Distribution (50% seeking cost-optimization 50% seeking time-optimization).

Index	Resource	Average	Total Job	Total	Total
	/ Cluster	Resource		Job Ac-	Job Re-
	Name	Utiliza-		$\operatorname{cepted}(\%)$	jected(%)
		tion $(\%)$			
1	CTC	36.71	417	98.32	1.678
2	KTH SP2	32.132	163	98.15	1.875
3	LANL	56.22	215	81.86	18.83
	CM5				
4	LANL	40.64	817	91.67	8.32
	Origin				
5	NASA	37.22	535	100	0
	iPSC				
6	SDSC	39.30	189	99.4	0.59
	Par96				
7	SDSC	79.16	215	76.2	23.7
	Blue				
8	SDSC	65.18	111	66.66	33.33
	SP2				

Table 2. Workload Processing Statistics (without Federation - Independent Processing/Resource)

Table 3. Workload Processing Statistics (With Federation)

Index	Resource	Average	Total	Total	Total	No. of	No. of	No. of
	/ Cluster	Resource	Job	Job Ac-	Job	Jobs	Jobs	Re-
	Name	Utiliza-		$\operatorname{cepted}(\%)$	Re-	Pro-	Mi-	mote
		tion $(\%)$			jected(%	$_{6}$)cessed	grated	jobs
						Lo-	to	pro-
						cally	Feder-	cessed
							ation	
1	CTC	85.85	417	100	0	383	34	80
2	KTH SP2	96.50	163	100	0	118	45	44
3	LANL	64.19	215	99.06	0.093	164	49	35
	CM5							
4	LANL	59.61	817	98.89	1.10	769	39	38
	Origin							
5	NASA	44.16	535	100	0	401	134	69
	iPSC							
6	SDSC	69.50	189	100	0	175	14	30
	Par96							
7	SDSC	64.55	215	100	0	130	85	57
	Blue							
8	SDSC	78.80	111	100	0	62	49	96
	SP2							

(3) All users seek time-optimization.

The budget and deadline distribution for the user having the job J_i , seeking cost-optimization is given by $c_{budget} = processing cost on(J_i, R_{c_m})$ (cost of executing the job J_i on the resource R_{c_m}), m < n such that

$$R_{cost}(R_{c_m}) = \frac{\sum_{i=1}^{n} (R_{cost}(R_{c_i}))}{n}$$
(29)

where n is the total number of resources in the federation.

 $t_{deadline} = execution time on(J_i, R_{c_m})$ (Execution time of the job J_i on the resource R_{c_m}), m < n, such that

$$R_{power}(R_{c_m}) = min(R_{power}(R_{c_i})) \quad i = 1...n$$

$$(30)$$

where n is the total number of resources in the federation.

The budget and deadline distribution for the user having the job J_i , seeking time-optimization is given by $c_{budget} = processing coston(J_i, R_{c_m})$ (cost of executing the job J_i on the resource R_{c_m}), m < n, such that

$$R_{cost}(R_{c_m}) = max(R_{cost}(R_{c_i})) \quad i = 1...n$$

$$(31)$$

where n is the total number of resources in the federation.

 $t_{deadline} = execution time on(R_{c_m})$ (Execution time of the job J_i on the resource R_{c_m}), m < n, such that

$$R_{power}(R_{c_m}) = \frac{\sum_{i=1}^{n} (R_{power}(R_{c_i}))}{n}$$
(32)

where n is the total number of resources in the federation.

In experiment-3, we measured the computational economy related behavior of the system in terms of 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. Total incentive earned by different resource owners with varying user population profile can be seen in (refer to Fig.(9)). Result shows that the owners (across all the resources) got more incentive when users sought time-optimization (Total Incentive 1.79E+09 Grid Dollars) (scenario-3) as compared to cost-optimization (Total Incentive 1.57E+09 Grid Dollars) (scenario-1). During time-optimization, we observed that there was a uniform distribution of the jobs across all the resources (refer to Fig.(8)) and every resource owner got some incentive. While during cost-optimization, we observed non-uniform distribution of the jobs in the federation (refer to Fig.(8)). We observed that some resource owners do not get any incentive (e.g. CTC, KTH SP2, NASA iPSC and SDSC SP2). This can also be observed in their resource utilization statistics (Fig.(8)) which indicates 0% utilization. These resources offered faster (response time) services but at a higher price. This is worst case scenario in terms of resource owner's incentive across all the resources.

This also indicates imbalance between the resource supply and demand pattern. As the demand was for the cost-effective resources as compared to the faster one, so these faster but expensive resources remained underutilized. All the jobs in this case were scheduled on other resources (LANL CM5, LANL Origin, SDSC Par96 and SDSC Blue), as they provided cost-effective solution to the users. With even user population distribution (during scenario-2) all the resource owners across the federation got incentive (Total Incentive 1.77E+09 Grid Dollars) and had better resource utilization (Fig.(8)). This scenario shows balance in the resource supply and demand pattern. Thus, we conclude that resource supply (No. of resource providers) and demand (No. of resource consumers and QoS constraint preference) pattern determines the resource owners overall incentive and the resource usage scenario.

We also measured the end-users QoS satisfaction in terms of average response time and average budget spent under two different optimization scenario (cost and time). We observed that end-users got better average response time (Fig.(10)) when they sought time optimization (scenario-3) for their jobs as compared to cost-optimization (scenario-1). At LANL Origin the average response time for the users was 6243.6 simulation seconds (scenario-1) which reduced to 4709.4 during time-optimization. The end-users spent more budget in case of time-optimization as compared cost-optimization (refer Fig.(11)). This shows that users get more utility for their QoS constraint parameter response time, if they are ready to spend more budget. Thus, we conclude that in user-centric resource allocation mechanism users have more control over the job scheduling activities and they can express their priorities in terms of QoS constraints.

We based rest of our experiments including experiment-4, experiment-5 and experiment-6 on the synthetic workload.

5.5. Experiment-4 (The affect of economic models on the cluster owner's overall profit)

In this experiment we evaluate how the profit of the cluster owners and the overall resource utilization varies with the pricing policy i.e. as they quote with different price. We performed this experiment with three clusters having configuration as shown in table-IV.

17



Fig. 8. Average Resource Utilization (%) Vs. Resource Name

Id	Cpu-type	Os-type	Secondary	Primary	Libs	MIPS	Nodes	Price
Cluster-1	Intel	Linux	20 GB	512MB	Gnu	600	5	Random (38)
Cluster-2	Intel	Linux	20 GB	512MB	Gnu	200	5	1
Cluster-3	Intel	Linux	20 GB	512MB	Gnu	300	5	2

Table 4. Experiment-4 Resource setup

At all three clusters we created heterogeneous set of user population having different optimization goals for their job.

At all three clusters heterogeneous set of user population having different constraint optimization preferences. initial User Population (60 Average Job Size 12000 MI) :

 $15 \leq \mathrm{budget} \leq 110 \; (\mathrm{Grid} \; \mathrm{Dollars}) \; , \; 25 \leq \mathrm{deadline} \leq 75 \; (\mathrm{simulation \; units})$

25% : Cost-Optimization

65% : Time-Optimization

 $10\%: \mbox{Cost-Time Optimization}$



Fig. 9. Total Incentive (Grid Dollars) Vs. Resource Name

We used the same user population in all our experiments but varied the price of the most powerful cluster which has MIPS rating of 600. We vary the price through 3,4,5,6,7, and 8 while keeping the resource price of other clusters fixed at 2 for cluster with MIPS rating 300 while 1 for the cluster having MIPS rating 200. The results of this simulation run in terms of total earnings and total jobs executed is shown in Fig.(12) and Fig.(13).

Initially when cluster-1 quotes with cost-factor 4, it executes 65% of total jobs while earning around 1400 grid-dollars. As this value is increased to 4 and 5, although less percentage of total jobs are executed at this cluster, but its earning increases due to high cost-factor and there is still appropriate demand for this resource type in the user population who have sufficient budget and they have opted for Time-Optimization strategy i.e. faster response time. But as this cost-factor is increased beyond 6 to value 7 and, the earnings of this cluster decreases considerably. This due to fact that those user seeking faster response time run out of their budget, so they cant get their job executed on the most powerful resource instead it gets executed on second most powerful cluster i.e. having MIPS rating 300 and which is offering the resources at affordable price. In this simulation the user's seeking cost-optimization for their job always get their job done on cluster-2 so, this cluster gets same number of jobs to execute while there is subsequent shift of jobs from cluster-1 to cluster-3. This signifies that the cluster owners get more earning for their resources if they offer them within reasonable price limit, as with subsequent increase in the price the demand for that resource may decrease considerably which leads to loss rather than profit. This may further lead to large number of user jobs getting rejected due to unsatisfied constraints, thus degrading QoS indicator for the system.



Fig. 10. Average Response Time (Simulation Units) Vs. Resource Name

Id (c_1, c_2, c_3)	Cpu	Os	S^* (GB)	RAM (MB)	Libs	MIPS	Nodes	Price
(1,11,21)	Intel	Linux	$20,\!10,\!10$	$512,\!512,\!256$	Gnu	600,200,360	$5,\!4,\!4$	4,1,2
(2,12,22)	i586	Linux	$10,\!20,\!20$	$256,\!256,\!512$	Gnu	500,220,370	4,3,2	5, 1.5, 2.4
(3,13,23)	i686	Linux	$10,\!10,\!10$	256, 256, 256	Gnu	700,235,380	3,3,2	6, 1.3, 2.5
(4, 14, 24)	Intel	solaris	$10,\!20,\!20$	$512,\!512,\!512$	Gnu	500,230,340	2,2,5	7,1.4,2.3
(5,15,25)	macintosh	macos	$20,\!10,\!10$	$256,\!256,\!256$	Gnu	700,200,370	4,4,4	6, 1.4, 2.6
(6, 16, 26)	Intel-P	WinXp	$10,\!20,\!20$	$512,\!512,\!256$	Vs6	800,230,330	2,3,2	5, 1.4, 2.4
(7, 17, 27)	macintosh	macos	$10,\!10,\!10$	$512,\!256,\!256$	mpi	700,200,300	4,5,2	5,1,2
(8,18,28)	alpha	Linux	20,20,20	$256,\!512,\!512$	Gnu	500, 255, 320	4,3,3	5, 1.3, 2.2
(9,19,29)	alpha	Linux	20,20,20	$256,\!512,\!512$	mpi	800,240,330	2,4,3	7,1.7,2.2
(10,20,30)	Intel	WinXp	10,10,10	512,256,256	.Net	700,260,350	3,4,2	6, 1.3, 2.4

Table 5. Experiment-5 Resource setup

5.6. Experiment-5 (The affect of QoS parameters on the service utility of the system)

In this experiment, we measured how different economic scheduling strategy affects the end-users QoS in terms of response time and budget spent. We simulated with 10 clusters in the federation, with user population spanning over all the clusters having different optimization constraints for their job. These users have varying job length starting with 12k to 24k. The table-5 and 6 depicts the experiment setup.



Fig. 11. Average Budget Spent (Grid Dollars) Vs. Resource Name

Job-Size(MI)	Cluster	Cpu	Os	S^* (GB)	RAM (MB)	Libs
12000	3	intel	linux	.2	64	gnu
24000	8	i586	linux	.1	32	gnu
16000	6	i686	linux	.2	64	gnu
18000	7	intel	solaris	.1	32	gnu
19000	9	intel-P	winxp	.1	64	vs6
20000	10	macintosh	mac-os	.2	32	gnu
22000	1	alpha	linux	.1	32	gnu
14000	4	macintosh	mac-os	.2	64	mpi
15000	5	alpha	linux	.1	56	mpi
18000	2	intel	winxp	.2	64	.Net

Table	6	Experiment_4	Ioh	setun
Table	0.	DAPOINTONO-T	000	beuup

 $^{\mathrm{a}}$

User Population (200, different optimization for 1 user/cluster monitored, average job size 12000 MI) $15 \leq budget \leq 150$ (Grid Dollars) , $25 \leq deadline \leq 120$ (simulation units)



Fig. 12. Total Earnings (Grid Dollars) Vs. Quote (Cost Factor)

We performed the experiments for same set of users but varying their optimization strategy while modifying the deadline and time constraints accordingly. Fig.(14) and Fig.(16) shows this experiment results.

We have also indicated the user specified value of deadline and budget constraints on the plots with the experiment results. For example, In cost-optimization user-4 spends 14 (specified budget constraint:16) Grid-Dollar getting 71 time units as response time (specified time constraint: 75 time units) whereas the same user spends 70 Grid Dollars while getting 21 time units as response time in case of time-optimization. It can be observed from the graph that a federation users get better response time in case of Time-Optimization strategy but they end up spending more budget as compared to Cost-Time and Cost Optimization strategies. Fig.(15) shows the plot of average response time along with standard deviation in all three optimization strategies. Fig.(17) shows the plot of average budget spent along with standard deviation in all three optimization strategies for different users.

5.7. Experiment-6 (System's acceptance rate with varying resource consumer size)

In this experiment, we measure the how QoS indicator varies with user population size (cluster wide), while maintaining a constant system size of 10 cluster resources. The table-7 shows the experiment setup.

At all three clusters heterogeneous set of user population having different constraint optimization preferences. initial User Population (Average Job Size 12000 MI) :

 $15 \leq \text{budget} \leq 110 \text{ (Grid Dollars)}$, $25 \leq \text{deadline} \leq 75 \text{ (simulation units)}$

37.5%: Cost-Optimization

22



Fig. 13. Total Jobs Executed Vs. Quote (Cost Factor)

Id	Cpu	Os	Secondary (GB)	RAM (MB)	Libs	MIPS	Nodes	Price
Cluster1	Intel	Linux	20	512	Gnu	600	3	4
Cluster2	Intel	Linux	20	512	Gnu	500	4	5
Cluster3	Intel	Linux	20	256	Gnu	500	3	5
Cluster4	Intel	Linux	20	512	Gnu	400	5	4
Cluster5	Intel	Linux	20	256	Gnu	250	3	1
Cluster6	Intel	Linux	10	256	Gnu	200	3	1
Cluster7	Intel	Linux	10	256	Gnu	250	5	1.5
Clsuter8	Intel	Linux	10	256	Gnu	150	3	1
Cluster9	intel	Linux	10	512	Gnu	300	4	2
Clsuter10	Intel	Linux	20	256	gnu	400	3	3

Table 7. Experiment-6 resource setup

37.5%: Time-Optimization

25%: Cost-Time Optimization

Fig.(18) shows this result for this experiment. For user population size of 200, we observed that about 81% of users had their constraints satisfied. This shows that system is provding good QoS to the end users, therefore QoS indicator for this system state is 81%. But as we increased the user population size to 1000, there was a sharp decrease in the total number of jobs that were accepted, approximately 51%. This indicates



Fig. 14. Response Time (Time Units) Vs. User ID (Federation Wide)

the degradation in the QoS indicator of the system. Further for user population size of 5000 we found that about 21% of the jobs were accepted. This experiment shows that with the increase in total number of end-users the performance of the system degrades considerably. We conclude that performance of a resource allocation system is determined by its QoS indicator and for a efficient system this parameter should have small degradation even with the increase in the resource consumer population.

6. Related Work

Grid resource management and scheduling has been investigated extensively in the recent past (Apples [8], NetSolve [14], Condor [30], LSF [2], SGE [24], Punch [28], Legion [15]). In this paper, we mainly focus on multi-clustering systems that allow coupling of wide area distributed clusters. We also briefly describe computational economy based cluster and Grid systems as we draw inspiration from them.

Load Sharing Facility (LSF) [2], is a very popular commercial batch queuing system which mainly supports campus grids. It focuses towards coupling of various local clusters for example departmental clusters under same administrative domain. It has the ability to run parallel jobs through the use of parallel virtual machine (PVM). Recently it has been extended to support multi-cluster environment by enabling transparent migration of jobs from one cluster to another. Although resource allocation strategy of LSF includes various priorities and deadlines mechanism, still it does not provide any mechanism for end users to express their valuation of resources and QoS constraints. Our *Grid-Federation* addresses this issue through user-centric resource allocation mechanism, which enable users to have better utility and control for their application scheduling.

Sun Grid Engine (SGE) [24] is a cluster resource management system developed by Sun Micro systems. The SGE enterprise edition allows the users to create campus Grid of clusters by combining two or more



Fig. 15. Time(Simulation Units) Vs. User ID (Federation Wide)

clusters in the local enterprise network. Each of these clusters is managed by SGE master manager. It has got a policy module which defines proportional based sharing of resources to the users of campus Grid, which in turn determined by the respective share of the user's cluster in the global share space. The users are assigned Tickets, which are like user's pass to use the campus Grid resources. They also get incentive for preserving their tickets during low computation period by getting more access tickets when they need more computational power. This policy is quite flexible depending on resource usage scenario and suited only to campus Grid environment under same administrative domain. It is not very useful for environment that consists of various resource owners with different resource sharing policies and resource consumers with different objective functions and QoS constraints. Our system supports policy based resource sharing where a resource owner can define how, what or when to share a resource and end user's can express their own resource usage scenario.

Condor [30] is a distributed batch system developed to execute long-running jobs on workstations that are otherwise idle. The emphasis of Condor is on high-throughput computing. Condor presents a single system view of pool of multiple distributed resources including cluster of computers, irrespective of their ownership domain. It provides a job queuing mechanism, scheduling policy, priority scheme, job check-pointing and migration, remote system calls, resource monitoring and resource management facilities. Scheduling and resource management in Condor is done through matchmaking mechanism [33]. Recently Condor has been extended to work with globus, the new version is called Condor-G, which enables creation of global Grids and designed to run jobs across different administrative domains. In contrast, we propose a more general scheduling system that views multiple clusters as cooperative resources that can be shared and utilized based on computational economy model of resources.

Nimrod-G [4] is a RMS system for wide-area parallel and distributed computing platform called the Grid. The Grid enables the sharing and aggregation of geographically distributed heterogeneous resources

25



Fig. 16. Budget Spent (Grid Dollars) Vs. User ID (Federation Wide)

such as computers (PCs, workstations, clusters etc.) software and scientific instruments across the Grid and presents them as a unified integrated single resource that can be widely used. Nimrod-G serves as a resource broker and supports deadline and budget constrained algorithms for scheduling task-farming applications on the Grid. It allows the users to lease and aggregate resources depending on their availability, capability, performance, cost, and users QoS constraints. The resource allocation mechanism and application scheduling inside Nimrod-G does not take into account other brokering system currently present in the system. This can lead to over-utilization of some resources while underutilization of others. To overcome this, we propose a set of distributed brokers having a transparent co-ordination mechanism, hence enabling cooperative resource sharing and allocation environment.

Libra [35] is a computational economy based cluster-level application scheduler. This system demonstrates that the heuristic economic and QoS driven cluster resource allocation is feasible since it delivers better utility than traditional system-centric one for independent job model. Existing version of Libra lacks the support for scheduling jobs composed of parametric and parallel models, and does not support inter-cluster federation.

Alchemi [31] is .Net based desktop grid computing platform. The main features of alchemi includes Internet-based clustering of window-class desktop machines, dedicated/non-dedicated resource sharing mode and file object based grid job model to enable legacy based applications. This allows trivial hierarchical coupling of various cluster resources in the Internet environment where master manager co-ordinates the application scheduling related activities with other managers that basically work as a dedicated/non-dedicated executors. It provides a application programming interface for the end-users to create grid applications. Like condor it presents a single system view of various resources including desktops, window-based clusters. In contrast we propose a scheduling system in which each resource manager co-ordinates with other resource manager, at the same level of ownership hierarchy not as a dedicated/non-dedicated executors, and perform



Fig. 17. Grid Dollars Vs. User ID (Federation Wide)

utility based resource allocation and hence enabling true policy based resource sharing.

REXEC [19] is remote execution environment for a campus-wide network of workstations, which is part of Berkeley Millennium Project. At command line, the user can specify the maximum credits per minute that he is willing to pay CPU time. The REXEC client selects a node that fits the user requirements. REXEC allocates resources to user jobs proportional to the user demands. It offers a generic user interface for computational economy on clusters, not a large scale scheduling system. It allocates resources to user jobs proportionality of the user valuation irrespective of their job needs, so it is more towards user centric type.

PBS [9] is flexible, POSIX compliant batch queuing and workload management system originally developed bu Verdian Systems for NASA. The purpose of PBS is to provide additional controls over initiating scheduling execution of batch jobs, and to allow routing of these jobs between different hosts. The default scheduler in PBS is FIFO whose behavior is to maximize the CPU utilization. That is, it loops through the queued job list and starts any job that fits in the available resources. However, this effectively prevents larges jobs from ever starting. To allow large jobs to start, this scheduler implements a "starving jobs" mechanism. This method may work for some situations, but there are certain circumstances where this course of action does not yield the desired results. New alternative schedulers that can be used with PBS have also been developed. Maui is one such advanced batch scheduler with a large feature set, well suited for high performance computing(HPC) platforms. It uses aggressive scheduling policies to optimize resource utilization and minimize job response time. It simultaneously provides extensive administrative control over resources and workload allowing a high degree of configuration in areas of job prioritization, scheduling, allocation and reservation policies. Maui also have a advance reservation infrastructure allowing sites to control exactly when, how and by whom resources are user.



Fig. 18. Total Job Accepted (Percentage) Vs. User Population Density (Federation Wide)

7. Conclusion And Future Work

In this report we proposed a new computational economy driven large scale scheduling system called gridfederation. The results of resource allocation algorithm indicates that our proposed framework leads to better overall utilization of cluster resources and it enhances the realization of objective function of resource owners and utility QoS constraints of resource consumers. We described how the variation in the objective functions of resource owners affect their profit and it may lead to degradation of the overall QoS indicator of the underlying system. We also presented a new QoS level indicator for grid systems. The results of the resource allocation algorithm indicates that resource supply and demand distribution and end-user quality of service constraints determines the actual QoS indicator of a resource allocation system. Our future work aims towards investigating co-ordinated QoS of service mechanism in the proposed framework and measuring the network complexity of such a system with large population density of resource providers and consumers. We also intend to look into new QoS constraint based algorithms for scheduling jobs containing parallel applications like MPI or PVM.

References

- 1. Parallel Workload Trace, http://www.cs.huji.ac.il/labs/parallel.
- 2. Platform, http://www.platform.com/products/wm/LSF.
- 3. J. H. Abawajy and S. P. Dandamudi. Distributed hierarchical workstation cluster co-ordination scheme. (PAR-ELEC'00) August 27 - 30, Quebec, Canada, 2000.
- 4. 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.
- 5. D. Aderson, J. Cobb, E. Korpela, L. Matt, and D. Werthimer. SETI@HOME: An experiment in public-resource computing. *Communications of the ACM, Vol. 45 No. 11, ACM Press, USA*, 2002.
- 6. 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 Internation? Parallel and Distributed Processing Symposium (IPDPS 2003), IEEE Computer Society Press, USA, April 22-26 Nice, France, 2003.

- 7. A. Artur and X. Zhichen. Scalable, efficient range queries for grid infromation services. 2nd International Conference on Peer-to-Peer Computing, 2002.
- 8. F. Berman and R. Wolski. The apples project: A status report. Proceedings of the 8th NEC Research Symposium, Berlin, Germany, 1997.
- 9. 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.
- R. Buyya, D. Abramson, and J. Giddy. An economy driven resource management architecture for global computational power grids. Proceedings of the International Conference in Parallel and Distributed Processing Techniques and Applications (PDPTA 2000), June 26-29, Las Vegas, USA, CSREA Press, USA, 2000.
- 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.
- 12. 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.
- 13. 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.
- H. Casanova and J. Dongara. Netsolve: A network server solving computational science problem. International Journal of Supercomputing Applications and High Performance Computing;11(3); Pages:212-223, 1997.
- 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.
- 16. J. Chase, L. Grit, D. Irwin, J. Moore, and S. Sprenkle. Dynamic virtual clusters in a grid site manager. In the Twelfth International Symposium on High Performance Distributed Computing (HPDC-12), June, 2003.
- 17. G. Cheliotis, C. Kenyon, and R. Buyya. *Grid Economics: 10 Lessons from Finance*. Peer-to-Peer Computing: Evolution of a Disruptive Technology, Ramesh Subramanian and Brian Goodman (editors), Idea Group Publisher, Hershey, PA, USA. (in print), 2004.
- M. Chetty and R. Buyya. Weaving computational grids: How analogous are they with electrical grids? Computing in Science and Engineering (CiSE), The IEEE Computer Society and the American Institute of Physics, USA, July-August, 2002.
- 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.
- 20. A. B. Downey. Using queue time predictions for processor allocation. 3rd Workshop on Job Scheduling Strategies for Parallel Processing which took place in conjunction with IPPS., 1997.
- 21. I. Foster and C. Kesselman. The grid: Blueprint for a new computing infrastructure. Morgan Kaufmann Publishers, USA, 1998.
- I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. http://www.globus.org/research/papers.html, 2002.
- I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. International Journal of Supercomputer Applications, Vol. 15, No.3, 2001.
- 24. 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.
- 25. A. Iamnitchi and I. Foster. On fully decentralized resource discovery in grid environments. *International Workshop* on Grid Computing, Denver, CO, 2001.
- 26. IEEE. Ieee std 802.3. Technical report, IEEE, 2002.
- 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.
- 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.
- M. Li, X. Sun, and Q. Deng. Authentication and access control in p2p network. Grid and Cooperative Computing: Second International Workshop, GCC 2003, Shanhai, China, December 7-10, 2003.
- 30. J. Litzkow, M. Livny, and M. W. Mukta. Condor- a hunter of idle workstations. *IEEE*, 1988.
- 31. A. Luther, R. Buyya, R. Ranjan, and S. Venugopal. Peer-to-peer grid computing and a .net-based alchemi framework, high performance computing: Paradigm and infrastructure. 2004.
- D. Moore and J. Hebeler. Peer-to-Peer:Building Secure, Scalable, and Manageable Networks. McGraw-Hill Osborne, 2001.
- 33. R. Raman, M. Livny, and M. Solomon. Matchmaking: distributed resource management for high throughput computing. *High Performance Distributed Computing*, 1998. Proceedings. The Seventh International Symposium

- 30 on , 28-31 July, 1998.
- 34. R. Al-Ali F. Rana, D. Walker, S. Jha, and S. Sohail. G-QoSM: Grid service discovery using qos properties. Concurrency and Computation: Practice and Experience Journal, 16 (5), 2004.
- J. Sherwani, N. ALi, N. Lotia, Z. Hayat, and R. Buyya. Libra: An economy driven job scheduling system for clusters. Proceedings of 6th International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia'02), 2002.
- 36. Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for internet applications. *To Appear in IEEE/ACM Transactions on Networking*, 2002.
- 37. 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.
- 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.
- 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.