CS 6203

Advanced Topics in Database Management Systems

Report: P2P vs Grid Computing

By

Ooi Hong Sain

HT029804y

Acknowledgement

All figures are taken directly from the original works [buyya2000, buyya2002].

Abstract

P2P and Grid computing are emerging as a new paradigm for solving large-scale problems in science and engineering. Both share a lot of similarities in terms of sharing, selection, and aggregation of geographically distributed heterogeneous resources. They are also differing in terms of targeted users and resources. With the latest advancement of technologies, one may conjectures that both computing systems may converge at some point. In this paper, we briefly review both computing platforms and describe a framework that might allow both technologies to be converged together.

1. Introduction

Recently, two new approaches to distributed computing have gathered must attention of computing communities: P2P and Grid computing. Both claim to address the problem of managing the large-scale computation societies (Foster and Iamnitchi). As both share a lot of similarities, this motivated our study towards comparing P2P and Grid computing.

Grid (Foster and Kesselman 1999) and Peer-to-Peer (P2P) (Oram 2001) computing platforms enable the creation of *Virtual Organizations* through sharing, selection, and aggregation of geographically distributed heterogeneous resources—such as computers and data sources—for solving large-scale problems in science, engineering, and commerce. The resources in these environments are heterogeneous and geographically distributed. The management of these resources and scheduling in such a large-scale distributed environment is a complex task.

Both have different requirements in the resource types and targeted users. However, both appear to have the same final objective that is the aggregation and coordination of the use of large sets of distributed resources. Based on the initial study (Foster and Iamnitchi), we found that

1. both technologies are concerned with the same general problem, mainly the managing of resources shared within the virtual organizations,
2. both take the same general approach in solving the problem, namely the creation of overlay architecture,
3. each approach has their own advantages and disadvantages as the targeted resources and users are different,
4. both approaches are likely to be converged over time.

In this study, we first study the similarities and differences between Grid and P2P computing. In Section 2, we discuss the technological advancements that contribute to the emerging of both technologies. Both technologies are briefly reviewed and the comparisons are given. In Section 3, we discuss an economic framework that may allow both computing platforms to be converged together. This is follow by the realization of the framework in Section 4. We expressed our own view in the issue of convergence of P2P and Grid computing. Finally, we conclude at Section 6.

2. P2P and Grid Computing

This section presents an overview of P2P and Grid computing technologies. It discusses some of the important technological advances that have led to the emergence of P2P and Grid computing.

P2P and Grid computing are two examples of distributed network computing system. By distributed, we mean the computing systems are allocated at geographically distributed regions. For example, the World-Wide Grid testbed consisted of computer resources that located in five continents: Asia, Australia, Europe, North America, and South America.

A distributed *network computing (NC)* system is a virtual computer formed by a set of heterogeneous computers (including equipments and other resources) linked together by network. Thus NC is a large scale collection of computing system linked together with network. With the pervasiveness of the Internet, the distributed network computing has been scale up to new global level. Thus with proper applications and tools, it can be used as Internet-size cluster.

The last decade has seen a substantial increase in computer performance. This is mainly the result of faster hardware and more sophisticated software. Nevertheless, there are still a lot of problems in the areas of science and engineering, which cannot be effectively dealt with even using the latest supercomputers. For example, proteins fold very quickly, as fast as a millionth of a second. However it takes a day to simulate a nanosecond of folding process in modern computers. In real life, proteins fold on the tens of microseconds timescale, this means that it would takes 10000 CPU days to simulate the folding (30 CPU years)(Folding@Home). This type of large-scale scientific problems has motivated the great amount of work in utilizing the distributed resources for solving large-scale problems.

We first present the technological advancements that enable P2P and Grid computing. These technologies are mainly the widespread of the Internet, availability of powerful computers and high-speed network connections. This is followed by the brief discussion about P2P and Grid computing. This section ended with the comparison between these two computing platforms.

2.1 Technological Advancement

The idea of linking computer systems together to solve problem is not a net idea. Back in the early 1970s, when the computers are first linked by network, the idea of harnessing unused CPU cycles was born (WWW). For example, scientists at Xerox Palo Alto Research Center (PARC) developed a program called “worm” that routinely cruised over 100 linked computer. The program was used to distribute graphic images and to share computation for realistic computer graphics rendering.

With the success of the Internet together with the availability of powerful computers and high-speed network technologies as low-cost commodity components, the computing landscape start to change. These technology opportunities have led to the possibility of using wide-area distributed computers for solving large-scale problems. Since 1990, with the maturation and ubiquity of the Internet and Web technologies together with the powerful computers and high-speed networks, distributed computing scaled up to a new global level. The availability of powerful PCs and workstations, and high-speed networks (e.g., Gigabit Ethernet) enable the emergence of clusters for high performance computing (HPC). The combination of the Internet and the clusters within many organizations has prompted the exploration of aggregating distributed resources for solving large scale problems of multi-institutional interest. This has led to the emergence of computational Grids and P2P networks for sharing distributed resources. The Grid community is generally focused on aggregation of distributed high-end machines such as clusters, whereas the P2P community (e.g., SETI@Home(SETI@Home)) is looking into sharing low-end systems, such as PCs connected to the Internet and contents (e.g., exchange music files via Napster(Napster) networks).

2.2 P2P Computing

P2P is a class of applications that takes advantage of resources -- storage, CPU cycles, and human presence -- available at the edges of the Internet (Shirky 2000). One unique characteristic of P2P applications is that P2P nodes must operate outside the DNS system and have significant or total autonomy from central servers. This is because accessing these decentralized resources means operating in an environment of unstable connectivity and unpredictable IP addresses. Another characteristic is that it eliminates the need for servers and allows all computers to communicate and share resources as equals. This is what makes P2P distinctive.

The P2P acronym technically stands for **peer-to-peer**. TechWeb (TechWeb) defines P2P as:

> "**From user to user**. Peer-to-peer implies that either side can initiate a session and has equal responsibility. Peer-to-peer is a somewhat confusing term, because it has always been contrasted to a central system that initiates and controls everything. **But in practice**, **two users on a peer-to-peer system often require data from a third computer**. For example, the infamous Napster file sharing service was always called a "peer-to-peer network," but its use of a central server to store the public directory made it both centralized and peer-to-peer. The two major categories of peer-to-peer systems are **file sharing** and **CPU sharing**."

Based on the above definition, P2P applications can be classified into two categories: P2P network and P2P computing. P2P network defines a communication environment that allows all desktop and laptop computers in the network **to act as servers** and share their files with all other users on the network. P2P networks are quite common in small offices that do not use a dedicated file server. In such cases, only specific folders in each machine are made sharable for read access only (not write access).

One the other hand, P2P computing focuses on sharing CPU resources across a network so that all machines function as one large supercomputer. It allows unused CPU capacity in any of the machines to be allocated to the total processing job required. In a large enterprise, hundreds or thousands of desktop machines are sitting idle at any given moment. Even when a user is reading the screen and not typing or clicking, it constitutes idle time. These unused processing cycles could be put to use on large computational problems. Likewise, the millions of users accessing the Internet create trillions of wasted machine cycles every minute that could be put to other use (see SETI@HOME).

2.3 Grid Computing

The Grid takes its name from an analogy with the **electrical power grid** that provides consistent, pervasive, dependable, transparent access to electricity, irrespective of its source. The motivation for computational Grids was initially driven by large-scale, resource (computational and data) intensive scientific applications that require more resource than a single computer (PC, workstation, supercomputer, or cluster) could provide in a single administrative domain. A Grid enables the sharing, selection, and aggregation of a wide variety of geographically distributed resources including supercomputers, storage systems, data sources, and specialized devices owned by different organizations for solving large-scale resource intensive problems in science, engineering, and commerce. (Buyya 2002)

With grid computing, an organization can transform its distributed and difficult-to-manage systems into a large virtual computer that can be set loose on problems and processes too complex for a single computer to handle efficiently. The problems to be solved can involve data processing, network bandwidth, or data storage. The systems linked in a grid might be in the same room, or distributed across the globe; they might be running different operating systems on many hardware platforms; they might even be owned by different organizations. Regardless of the depth of a grid's resources, what is appears to all grid users is a very large virtual computer. (Computing)

The major purpose of a Grid is to gather resources to solve large-scale problems; the main resources grid computing is designed to give access to include (but are not limited to):

- Computing/processing power
- Data storage/networked file systems
- Communications and bandwidth
- Application software

Since the concept of putting grids into real-world practice is still relatively new, another good way to describe a grid is to describe what it isn't. The following entities are not grids:

- Cluster
- Network-attached storage device
- Scientific instrument
- Network

Each might be an important component of a grid, but by itself, doesn't constitute a grid.

2.4 Comparing Grid and P2P Computing

P2P and Grid computing share a lot of similarities, both allow the sharing, selection, and aggregation of geographically distributed resources (virtual organization). The main difference between P2P and Grid computing is that The **Grid community** is generally focused on aggregation of distributed **high-end** machines such as clusters, whereas **P2P community** is looking into sharing **low-end** systems, such as PCs connected to the Internet (e.g. SETI@Home) or contents (file sharing via Napster networks).

Current Grids provide many services to **moderate-sized communities** (mostly scientific communities) and emphasize the integration of substantial resources (high-end resources) to deliver **nontrivial qualities of service** within an environment of at least limited **trust**. In contrast, current P2P systems deal with **many more participants** (e.g., hundreds of thousands in Napster) but offer **limited and specialized services such as file-sharing.** P2P systems have been **less concerned with qualities of service**, and have made **few if any assumptions about trust**. This is mostly due to the dynamic nature of the networks; as each user can join and leave the system at any time (Foster and Iamnitchi).

3. Grid Architecture for Computational Economy (GRACE)

In this section, an economic framework for managing resources and scheduling applications in Grid computing environments is presented. The framework is proposed as a solution to the challenges in Grid computing environments. The motivation of such an approach is due to the success of economic models for exchanging and regulating goods, services, and resources used in the real world. Several real world economic models such as commodity market, posted prices, bargaining and tendering are discussed.

3.1 The requirements of economic-based approach

In addition to normal resources management problems such as site autonomy, policy extensibility, online control, and heterogeneous substrate, the economic-based approach also introduced new issues such as resource trading and quality of service-bases scheduling. Thus to address some of these issues, the framework must support the following:

- An information and market directory (publications)
- Models for representing value of resources (pricing)
- Economic models and negotiation protocols
- Regulatory agencies (mediators)
- Accounting, billing, and payment mechanisms
- Users' Quality of Services

To allow resource owners and consumers to express their requirements and facilitate the realization of their goals, the following mechanisms are required:

- **Value expression**
- **Value translation**
- **Value enforcement**

The value expression is a mechanism that allows both parties to express their requirements, valuations, and objectives. The value translation is a mechanisms used by the scheduling policies to translate the values into resource allocations. Finally, the enforcement of selection and allocation of differential services, and dynamic adaptation to changes in the availability at runtime is handled by the value enforcement mechanism.

For example, the users can specify the deadline and budget constraints along with optimization parameters [*value expression*]. Then the client application provides strategies for choosing appropriate resources [*value translation*] and dynamically adapt to changes in resource availability at runtime to meet user requirements [*value enforcement*]. On the other hand, the resource owners specify the prices to increase system utilization together with protocols that help them offer competitive services [*value expression*]. The Grid resource schedulers will allocate the resources [*value translation*] and allocate the resources during reserved time [*value enforcement*].

3.2 GRACE

GRACE was proposed around year 2000 to address the resource management challenges (Buyya, Abramson et al. 2000; Buyya, Abramson et al. 2001; Buyya, Stockinger et al. 2001). The challenges include site autonomy, heterogeneous substrate, policy extensibility, resource allocation or co-allocation, online control, resource trading, and quality of service based scheduling. Most of the problems are already solved in currently available Grid system, such as Globus (Foster and Kesselman 1997), Legion (Grimshaw and Wulf 1997), and Condor (Litzkow, Livny et al. 1988). The GRACE framework was proposed to address the two remaining issues: resource trading and quality of service-based scheduling. Furthermore, to reduce the duplication of works, GRACE uses existing technologies such as Globus, and Legion, and develops other services on top of these technologies.

The architecture of GRACE is given in Figure 1. The architecture is designed in such as way that it is generic enough to accommodate different real world economic models. The models are used for resource trading and determining the service access cost. The key components of GRACE include:

- Grid user with applications
    - Sequential, parametric, parallel, or collaborative applications
- User-level middleware
    - Programming environment
    - Grid Resource Brokers
- Core Grid Middleware
    - Services resources trading and coupling distributed wide are resources
- Grid Service Providers

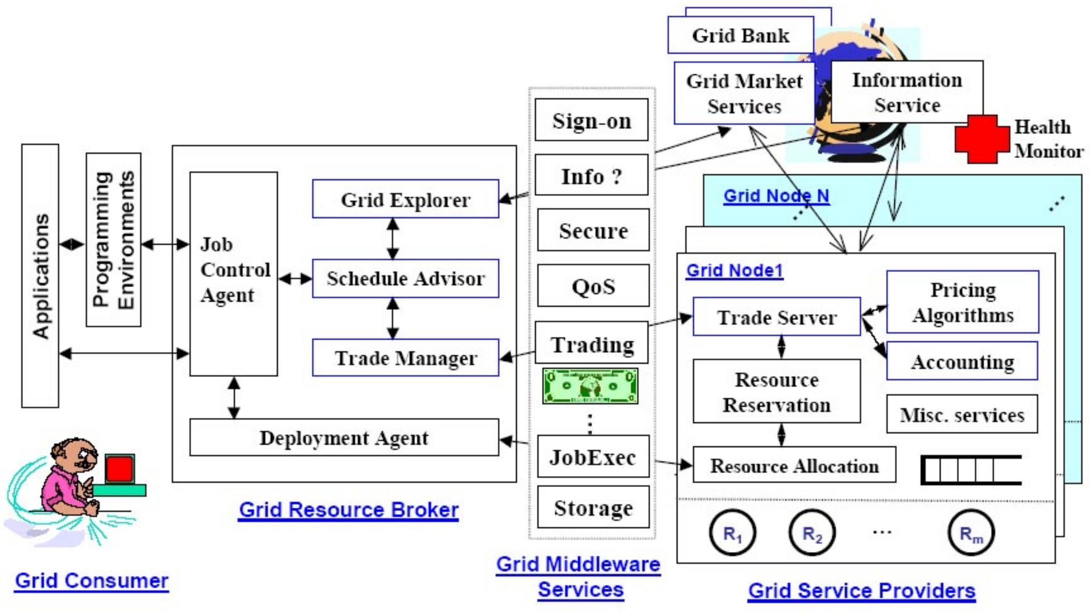

Figure 1. A framework of GRACE

There are two important players in GRACE, the resource owners and consumers. The resource owners are organizations or individuals that agreed to contribute their resources, such as personal computers, clusters, super-computers, storage systems, data sources, and other specialized devices. They are known as Grid Service Providers (GSPs) under GRACE. On the other hand, the resource consumers are those who utilized the resources to solve their problems. They are represented by Grid Resource Broker (GRB) which acts the consumer’s representative.

Both parties have their own expectations and strategies for being part of the environment. Basically, the resource consumers adopt the strategy of solving their problems based on their budget and deadline. Thus the consumers will choose the providers that best meet their requirements. The resource owners adopt the strategy of obtaining the best possible return on their investment. Thus they are more likely try to offer a competitive service access cost in order to attract consumers.

In order to allow both parties to express their requirements, some tools and mechanisms are needed. In GRACE, the consumers interact with GRB to express their requirements such as the budget and deadline. The budget defines the price that the consumers willing to pay for solving their problems, while the deadline defines the time frame by which they need the results. To resemble real world economic models, the negotiation protocols are also needed. This allows the consumers to trade between the deadline and budget requirements and steer the computations accordingly. The GSPs need tools for expressing their pricing schemes and mechanisms that can help them to maximize resources utilization and their profits.

3.2.1 Grid Resource Broker (GRB)

The resource broker acts as the middle man between the consumer and resources using middleware services. It presents the Grid to the consumer as a single and unified resource. This facilitates the adoption of previous technologies into GRACE framework. The following are major components in GRB:

- Job Control Agent (JCA)
- Schedule Advisor (Scheduler)
- Grid Explorer (GE)
- Trade Manager (TM)
- Deployment Agent (DA)

The GRB is responsible for resource discovery, resource selection, binding of software, data, and hardware resources, initiating computations, and adapting to the changes in Grid resources. These services are carried out by the components in GRB. The JCA is a persistent control engine responsible for managing a job through the system. Together with scheduler, it coordinates schedule generation, handles creation of jobs and maintains the job status. It interacts with the users/clients, scheduler and deployment agent. The scheduler is responsible for resource selection and job assignment with the help of scheduler through schedule generation. It is also responsible for resource discovery with the help of GE. The scheduler is used to ensure the user requirements are met. The GE is the main component responsible for resource discovery. It interacts with the Grid Information Server to retrieve and identify the list of authorized and available machines, and keeping track of the resources status. The TM works based on the requirements of the consumer. It uses the resource selection scheme generated by scheduler (based on user's requirements) to identify the resource access costs and negotiation protocols for trading with GSPs. The DA is responsible for the activation of jobs on the selected resources and updates the job status to JCA.

3.2.2 Grid Service Provider (GSP)

The GSPs specifically deal with the following components along with other services provided by Grid toolkits such as Globus and Legion:

- Grid Market Directory
- Grid Trade Server
- Pricing Policies
- Resource Accounting and Charging

The service providers publish their services through the GMD as normal businessmen publish their products and services at yellow pages.

3.2.3 Negotiation protocols

The negotiation protocols define the rules and format for exchanging commands between a trade manager and a trade server. The negotiation protocols play an important role in helping the resource owners and resource consumers to achieve their specific goals. Figure 2. shows a sample multilevel negotiation process when trading for the cost of resource access.

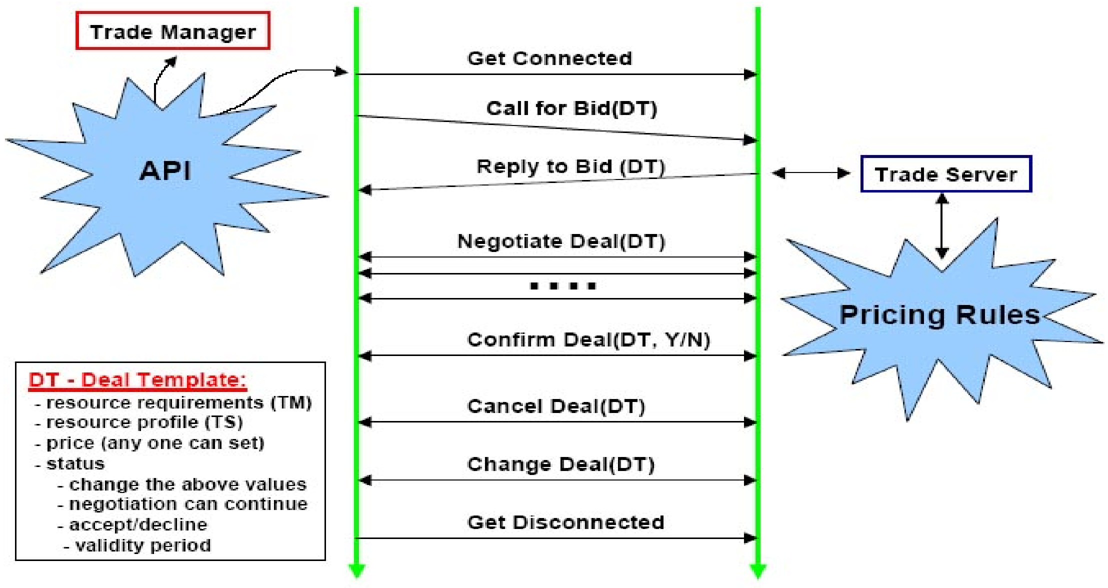

Figure 2. Negotiation Protocol

The Deal Template (DT) is used as a template for TM to specify resource requirements. The structure of the DT is given in Figure 2. The contents of DT can be changed during the negotiation process. The negotiation process between TM and TS continues until one of them indicates that its offer is over. Then the other party will decide whether to accept or reject the deal. If accepted, both work on what is specified in the DT.

3.3 Pricing

In an economic model, the resources consumed by the user applications need to be charged for usage fees. A simple pricing scheme such as fixed price model can be employed, but it may not work well when users place QoS demands that vary with application and time. Many works have been done in this area, the pricing schemes based on different parameter include:

- A flat price model
- Competitive economic model
- Usage timing, period and duration
- Demand and supply
- Loyalty of customers
- Bulk purchase

3.3.1 Services items to be charged

User applications have different resource requirements depending on the nature of the algorithms used in solving problems. Some applications are CPU intensive while others can be I/O intensive or combination. Some applications required large amount of memory or storage. These are all the items that can be chargeable. Therefore, in GRACE, the consumption of the following resources needs to be charged:

- CPU (including user time and system time)
- Memory
- Storage used
- Network bandwidth consumption
- Specialize devices
- Software and libraries accessed

Consumption of other resources can also be charged. The information about the services available and its corresponding charges can be obtained from the Grid Market Directory. The access to each of these entities can be charged individually or in combination. The pricing will be dependent on the strategies adopted by the service providers.

3.3.2 Payment Mechanisms

The resources consumed by the consumers need to be accounted and charged. Thus various payment mechanisms must be supported. The payment mechanisms supported by GRACE include:

- Prepaid
- Postpaid
- Pay as you go
- Grants based

The consumers can purchase resource access credits through any of the above schemes. Each GSP can maintain this by using system such as QBank or GridBank to mediate payment mechanisms. This approach reduces the great burden on the consumers and service providers in a large-scale Grid environment. Using this approach, the GRB inform the GSPs about the consumer GridBank account details for which they can charge directly or users can pay by other electronic cash systems such as:

- NetCheque (Neuman and Medvinsky 1995)
- NetCash (Medvinsky and Neuman 1993)
- Paypal (Paypal)

Such payment mechanisms satisfy the diverse requirements of consumers and service providers and can be easily integrated into GRACE.

3.4 Economic models in GRACE

GRACE is a generic economic framework that is capable of accommodating different models that are used in human economies. In this section, some models are discussed and their implementations are given. For each of the economic models, the economic model theory, its parameters and strategies are also presented.

The idea of applying economics to resource management in distributed systems such as Grid and P2P computing is not a new idea. Several researches have been done in this area, for example, Spawn (Waldspurger, Hogg et al. 1992), Popcorn (Nisan, London et al. 1998), and Java Market (Amir, Awerbuch et al. 1998). These works have been help in understanding the potential benefits of market-based systems. Unfortunately, many of them were limited to experimental simulations. Furthermore, the systems were implemented using monolithic approach, this make them hard to be scale up. Some expect the users to develop resource-aware applications explicitly for their platforms using their own programming interface (e.g., Spawn and Popcorn). As a consequent, developing applications for such platforms are difficult as users have to address both the application development and resource allocation issues concurrently.

This problem can be overcome by separating the application development and resource management issues and is used in GRACE.

3.4.1 Commodity Market

In commodity market model, resource owners specify their service price and charge users according to the amount of resource they use. The service price can either be flat or variable depending on the resource supply and demand. The service providers can adopt different strategies to increase the resource utilization, thus mixture of flat and variable price models can be used. In general economic model, the services are priced in such a way that there exists equilibrium between supply and demand.

In flat price model, the price is fixed for a certain period. It remains the same irrespective to the service quality and is not influenced by the supply and demand. On the other hand, in variable price model, the price changes very often based on supply and demand changes. When the demand increases or supply decreases, the service providers can increase the service prices until the supply and demand return to their equilibrium state again. Basically, the pricing schemes in a commodity market model can include:

- Flat fee
- Usage duration
- Subscription
- Demand and supply-based

The resource owners publish their prices and rules in the GMD service (as shown in Figure 3). This is similar real world businessmen publish their products and services through yellow pages. This is performed through the help of GTS.

The price specification may take the following form,
Price(owner_id, peak_price, offpeak_price, up_highdemand, down_lowdemand, holiday_price)

The owner_id is used to identify the resource owner, which may be same as Grid-ID. The price can be specified for peak period, say, between 9am to 6pm on working days and for off peak period. The price also be increased when there is high demand, or decreased when the demand is low. For example, the price can be reduced when the system load is less than 50% at any given time.

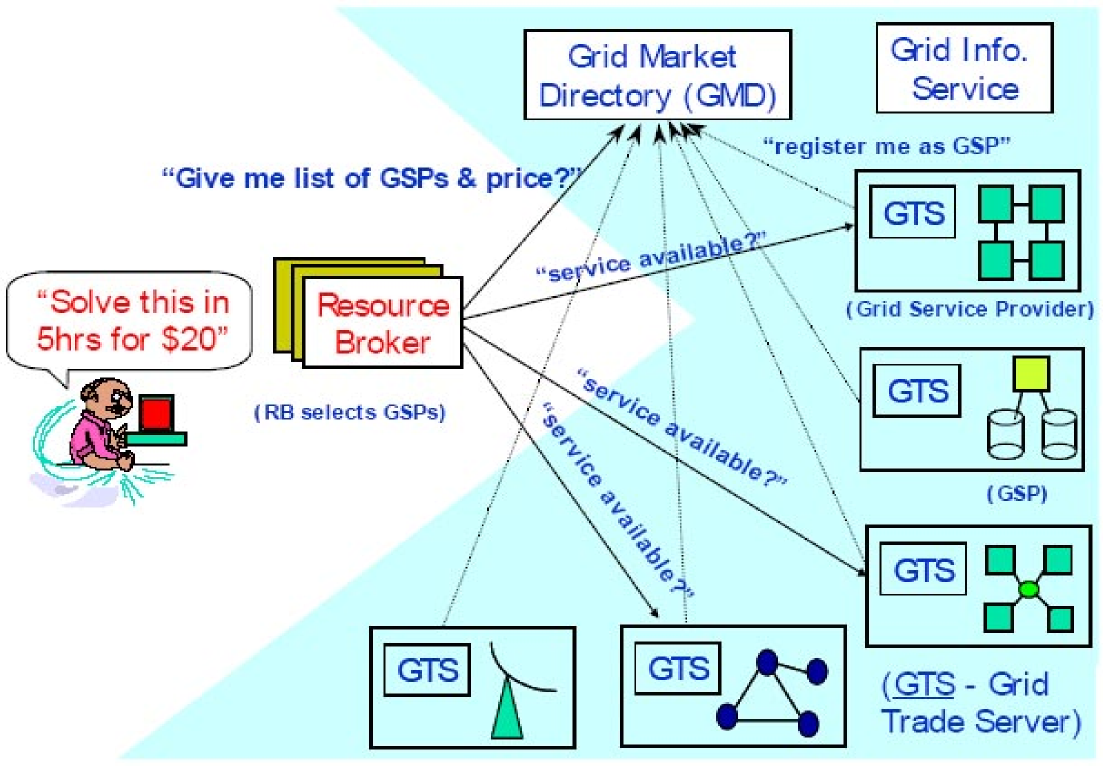

Figure 3. Commodity Market Model

To execute user applications on the Grid, the following steps are performed by the GRB:

1. The GRB identifies service providers (GSPs)
2. It identifies suitable resources and establishes their prices
3. It selects resources that meet user's objectives and requirements.
4. It uses resource services for job processing and issues payments as agreed.

3.4.2 Posted Price Model

The posted price model is similar to the commodity market model, except that special offer prices are used to attract consumers. This strategy can be used by new service providers to establish their market share or motivate users to consider using cheaper slots. In this model, the posted prices are used directly by GRB as they are generally cheaper compared to regular prices.

In general, the posted price offers will have usage conditions. For example, during holiday periods, demand for resources is likely to be limited. In this case, the GSPs can post tempting offers or prices to attract users to increase resource utilization.

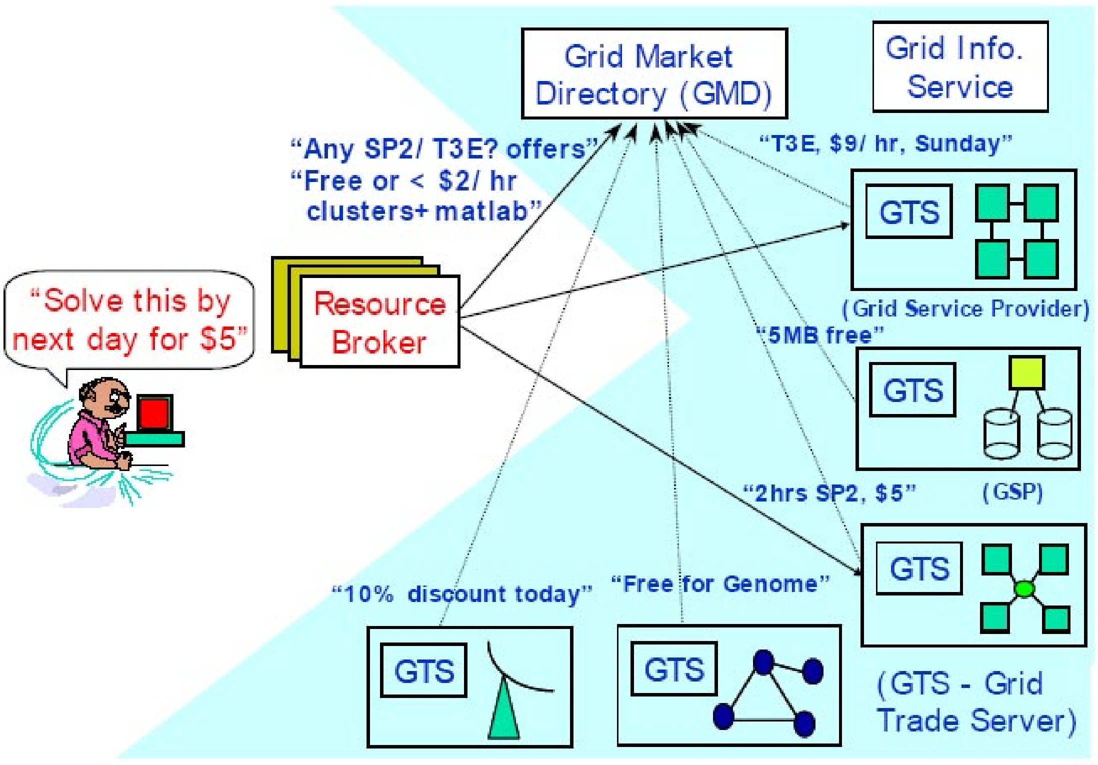

Figure 4. Posted Price Model

The steps in executing applications in posted price model are:

1. GSPs post their special offers and associated conditions in GMD
2. GRB looks at GMD to identify if any of these posted services available and fits its requirements
3. GRB enquires GSP for availability of the posted services
4. Other steps are similar to the steps used in commodity market model

3.4.3 Bargaining Model

In previous two models, the prices are fixed by the service providers. The consumers pay the access fees based on the agreed pricing scheme. There is no negotiation process in these models. In the bargaining model, GRB bargains with GSPs for lower access price and higher usage duration. Both parties have their own objectives and they negotiate with each other as long as their objectives are met. The process is illustrated in Figure 5.

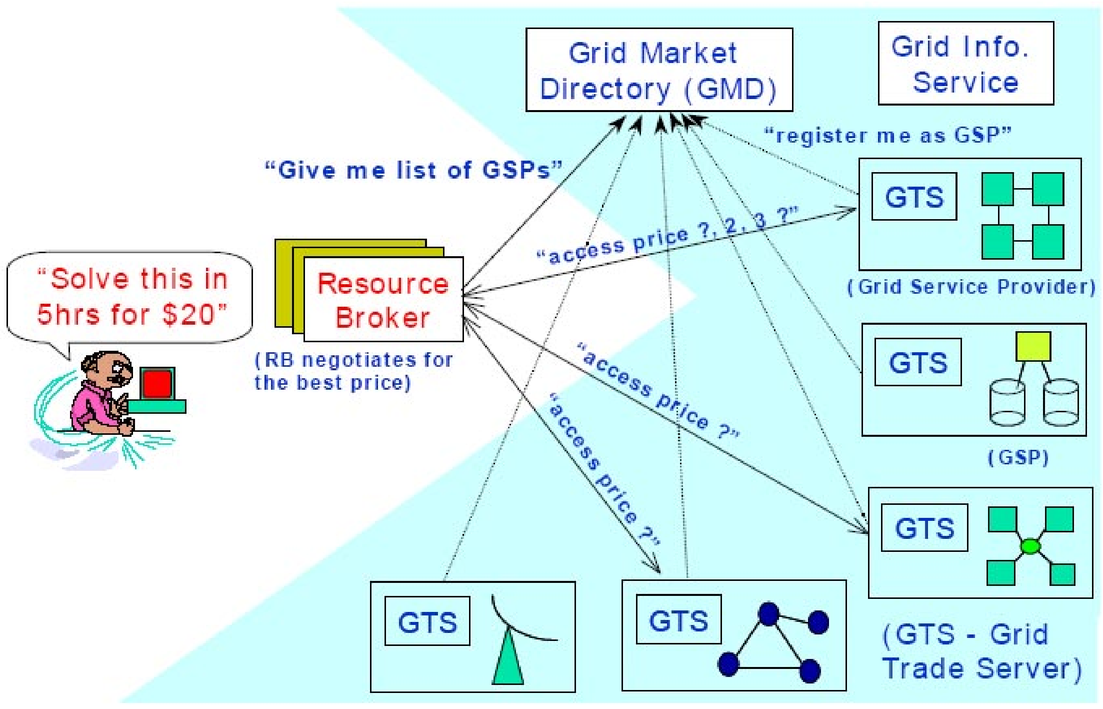

Figure 5. Bargaining Model

For example, the GRB might start with a very low price and GSPs with a higher price. They both negotiate until they reach mutual agreeable price or one of them decided to stop the negotiation. The negotiation is depended on the consumer requirements, such as deadline and budget. In some situations, the GRB may take risk and negotiate for cheaper prices as much as possible and discharge some expensive machines. This will be resulted in low utilization of resources, so the corresponding GSPs might be willing to reduce their service prices to attract consumers instead of wasting the resources.

3.4.4 Tender/Contract-Net Model

Tender/Contract-Net model is based on the real world tender and contract model. It is also one of the most widely used models for service negotiation in a distributed environment for solving problems. The main advantage of this model is that it helps in finding an appropriate service provider to work on a particular task. The process is given in Figure 6.

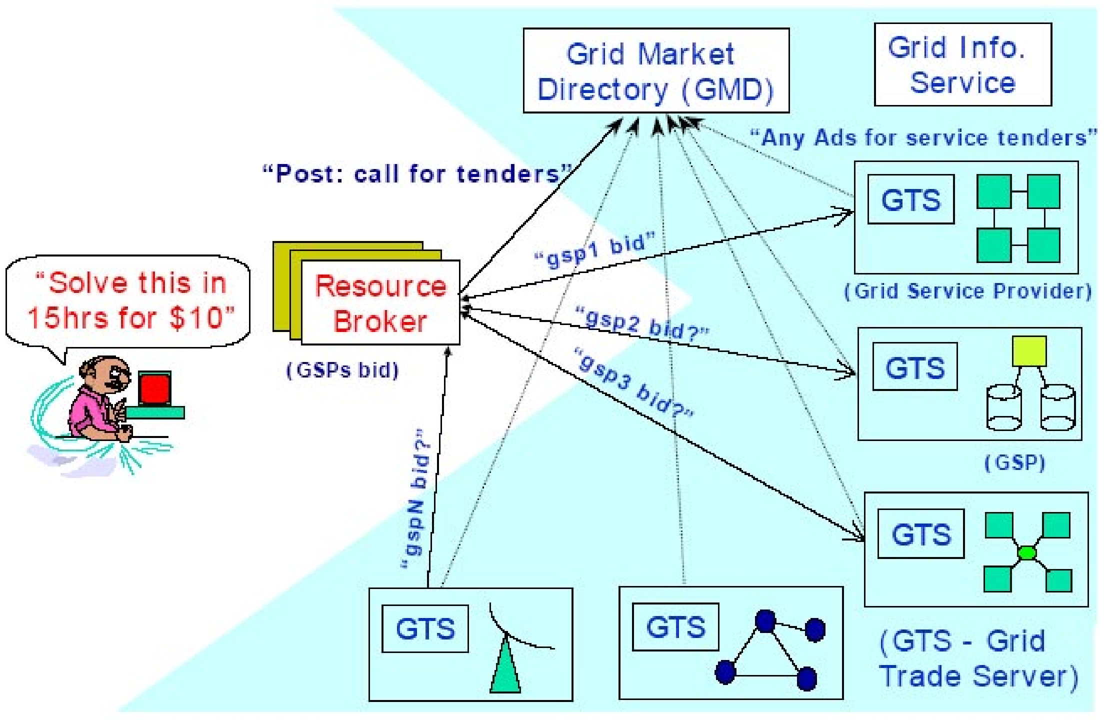

Figure 6. Tender/Contract-Net Model

In this model, the following steps are used by consumers to find suitable service provider (contractor):

1. GRB announces it requirements and invites GSPs for bids
2. Interested GSPs evaluate the announcement and respond by submitting their bids
3. GRB evaluates bids and awards the contract to most appropriate GSPs
4. Both parties communicate privately and utilizing the resources

On the other hand, the following steps are used by the service providers (contractors):

1. Receive tender announcements
2. Evaluate service requirements and its corresponding capability
3. Respond with bid
4. Deliver service if bid is accepted
5. Report results and charge the consumers according to agreed bid

Another advantage of this model is that if the selected GSP is unable to deliver a satisfactory service, then GRB can looks for services of other GSPs.

Due to the heterogeneity of Grid environments, this protocol has some disadvantages. For example, a task may be awarded to less capable GSP if a more capable GSP is busy at award time. Furthermore, the GSP may not be able to respond to the bid at the time if the resources are already occupied or the requirements are not appropriate.

3.4.5 Other Models

As mentioned earlier, GRACE is a generic framework that capable of accommodating different economic models. Other proposed economic models for GRACE include auction, bid-based proportional resource sharing model, cooperative bartering model and monopoly models.

The auction model supports one-to-many negotiation, between a service provider and many consumers. The auctioneer sets the rules of auction, which are acceptable to both the consumers and the providers, and the auction starts. In bid-based proportional resource sharing model, the percentage of resource share allocated to the user application is proportional to the bid value of a particular consumer. In cooperative bartering model, a cooperative computing environment is formed. Those who are contributing their resources to the environment can get access to the resources. In monopoly model, there exist cases where a single GSP dominates the market, for example, it is a single service provider of a special service. In this model, the consumers cannot influence the price of the service and have to use the price set by the GSP.

4. Nirmod-G Grid Resource Broker

This section presents the Nimrod-G Grid resource broker as an example of realization of GRACE. The architecture is generic enough to leverage services provided by various Grid middleware such as Globus, Legion, and Condor. This helps in reducing the development time. Furthermore, it also allows uniform access to diverse resources, managed by different Grid middleware.

Nirmod-G is a computational economy-based Grid resource management and scheduling system that supports deadline- and budget-constrained algorithms for scheduling parameter sweep applications (parameter studies) on distributed resources (Buyya, Abramson et al. 2000). It provides a simple declarative parametric modeling language for expressing parametric experiments. This allows one to create parameter-sweep applications. For example, the domain experts (application-specific experts) can easily create a plan for the application and use the Nirmod-G broker to handle all the issues related to resource managements and execution. The resource management and scheduling algorithms are based on economic principles (as a realization of GRACE framework).

4.1 Architecture

A diagram of high-level architecture and component of Nirmod-G is given in Figure 7.

The components include:

- A persistent task farming engine (TFE)
- A grid explorer
- A resource trading manager
- A schedule advisor
- A dispatcher and actuators
- Agents for managing execution of jobs

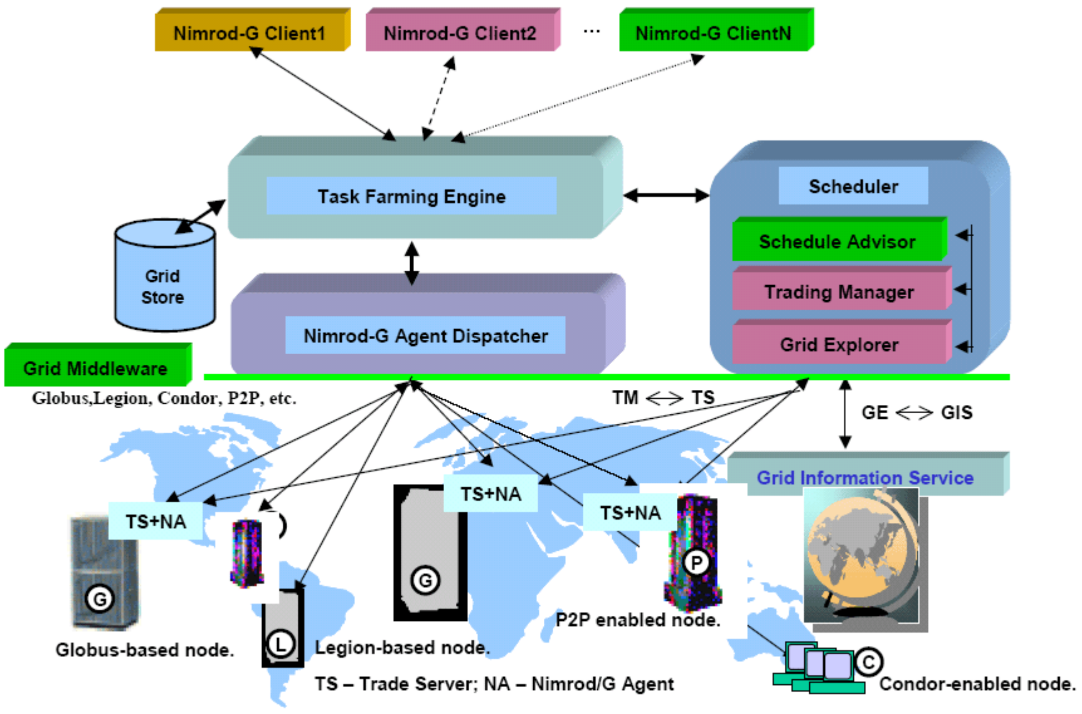

Figure 7. High-Level Diagram for Nirmod-G Architecture

The Nimrod-G resource broker is responsible for determining the requirements that a user application places on the Grid and performing resource recovery, scheduling, dispatching jobs, job execution and return the results to user application. The TFE is a persistent and programmable job control agent that manages and controls the user applications. It is responsible for managing the execution of parameterized application jobs. It coordinates resource trading, scheduling, data staging, execution and gathering results from remote Grid nodes to the user's home transparently. The programmable capability of TFE enables the creation and "plug-in" of user-defined scheduling policies and customization of the problem-solving environments.

The scheduler consists of a Grid explorer for resource discovery, a schedule advisor and a resource trading manager. It is responsible for resource discovery, resource trading, resource selection, and job assignment. The GE interacts with GMD to retrieve a list of authorized and available machines and their costs. The resource trading manager is responsible for keeping rack of resource status information and trades for resource access costs. The schedule advisor is backed with resource selection algorithm for selection of resources that meet the deadline and budget constraints.

The dispatcher and actuator are used for deploying agents on Grid resources. The dispatcher triggers appropriate actuators depending on middleware service to deploy agents on Grid resources and assign one of the resource-mapped jobs for execution. For example, a Globus-specific actuator is required for Globus resources.

The agent is responsible for setting up the execution environment on the selected resources. It transfers the code and data to the remote machine, starts the job on the assigned resource and returns the results back to the TFE. The agent also records the amount of resource consumed during the job execution. This helps the scheduler to evaluate the resource performance and change the schedule accordingly.

4.2 Scheduling Algorithms

The integration of computational economy into the scheduling algorithm greatly influences the selection of computational resources that meet the user requirements. The users should be able to submit their applications together with requirements. The scheduling algorithm should be able to process the application on the Grid on the user's behalf and try to complete the assigned work within a given budget and deadline.

In order for the scheduling algorithm to arrive at an optimal decision, various factors are needed to be considered. These factors include:

- Resource architecture, configuration, capability, and state
- Resource requirements of an application
- Free or available nodes
- Access speed
- Network bandwidth, load, and latency
- Reliability of resources and connection
- Application deadline
- Resource cost, and others

In Nirmod-G, the applications itself contain a large number of independent jobs operating on different data sets. A range of scenarios and parameters to be explored are applied to the program input values to generate different data sets. This resembles the SPMD (Single Program Multiple Data) computational model. The execution model essentially involves processing N independent jobs (same program but operates on different datasets) on M distributed computers.

The execution of the applications on distributed computers may appear straight forward, but complexity arises when deadline and budget constraints are applied to the scheduling algorithms. It is hard to guarantee service quality in such environments as the resources are shared, heterogeneous, geographically distributed and owned by different organizations having different policies. On top of these, the scheduling algorithms need to consider the changing load and resource availability conditions in the Grid in order to achieve performance and at the same time meet the deadline and budget constraints.

In Nimrod-G, three scheduling algorithms are used:

- Cost optimization
- Time optimization
- Conservative time optimization

Each algorithm works within the time and budget constraints. The role of deadline and budget constraints in these algorithms is given in Table 1.

Algorithms	Time	Cost
Cost optimization	Limited by deadline	Minimize
Time optimization	Minimize	Limited by budget
Conservative time optimization	Limited by deadline	Limited by budget

Table 1. Schedule Algorithm Based on Deadline and Budget Constraints

The time optimization scheduling algorithm tries to complete the job as quickly as possible, within the budget available. It is based on the following algorithm:

1. For each resource
    a) Calculate the next completion time for an assigned job
    b) Taking into account previously assigned jobs and job completion rate
2. sort resources by next completion time
3. assign one job to the resource which the cost per job is less than or equal to the remaining budget per job
4. repeat above steps until all jobs are performed

On the other hand, the cost optimization scheduling algorithm attempts to complete the job as economically as possible within the deadline. The algorithm is given as:

1. sort resources by increasing cost
2. for each resource in order
    a) assign as many jobs as possible to the resource
    b) make sure the assignment is not exceeding the deadline

The conservative time optimization scheduling algorithm tries to complete the job within the deadline and budget constraints. It tries to ensure that a minimum of “the budget-per-job” from the total budget is available for each unprocessed job. A description of the algorithm is as follows:

1. split resources based on cost per job (is less than or equal to the budget per job)
2. for the cheaper resources
    a) assign jobs in inverse proportion to the job completion time
3. for the dearer resources
    a) repeat all steps until all jobs are performed

5. Convergence of P2P and Grid Computing

Both P2P and Grid Computing share a lot of similarities, they both enable the sharing of resources through the creation of virtual organizations. We already briefly describe the main different between P2P and Grid computing. Nevertheless, there are some overlapped areas in P2P and Grid computing, this is especially true in the area of scientific computation. For example, the Folding@HOME is similar with other applications that simulate protein folding running in grid. The methods/platforms used might be different, but the approach is the same – solving large scale problem with collection of computing systems together.

With the advances of the technology developments, such as powerful personal computers and high-speed network connections, one may conjectures that at one point both P2P and Grid computing will converge together. This is the motivation behind this study. This might be possible if a proper framework and tools are available to integrate all currently available Grid middleware and P2P applications.

However, there will be areas that will only limit to particular system. For example, the illegal file sharing applications will likely be stand alone without converge with other systems. The developers of such systems will continue their efforts in developing new applications for this purpose. There will be other grid systems that will limited itself to certain groups of users as this may include highly advanced equipments and computing systems, hence trust and security are real issues. For example, the security agencies may employ their own Grid architecture that will well fit for their own purpose instead of using the open and general Grid utilities.

6. Conclusion

In this study, we briefly reviewed the two new emerging technologies for next generation computing, the P2P and Grid computing. They both share some similarities in terms of sharing the resources. At current stage, due to the targeted users and resources, there is a clear cut between these two technologies. However, as discussed above, with the advance of technology, the line is becoming unclear.

We presented a framework that allows both the P2P and Grid computing to be converged together. This framework is based on the economic models. The motivation behind this approach is to reward those who contribute to the resources. For P2P communities, the main reward is able to access files with minimum fees (music files such as mp3). Using this approach, we may encourage the P2P communities to share their computer cycles for solving large-scale problems by rewarding them. Furthermore, under this framework, different computing platforms such as Grid and P2P computing can be integrated together as shown in Figure 7.

However, no matter how successful the framework is, it is most likely that both platforms will not converge fully. There will be some specific areas that will remain specific to each of the computing platform.

Bibliography

Amir, Y., B. Awerbuch, et al. (1998). A Cost-Benefit Framework for Online Management of a Metacomputing System. Proceedings of 1st International Conference on Information and Computational Economy, Charleston, SC, USA.

Buyya, R. (2002). Economic-based Distributed Resource Management and Scheduling for Grid Computing. School of Computer Science and Software Engineering. Melbourne, Australia, Monash University. **PhD**.

Buyya, R., D. Abramson, et al. (2000). An Economy Driven Resource Management Architecture for Global Computational Power Grids. Proceedings of the 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000), Las Vegas, USA, CSREA Press.

Buyya, R., D. Abramson, et al. (2000). Nimrod-G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid. The 4th International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2000), Beijing, China, IEEE Computer Society Press, USA.

Buyya, R., D. Abramson, et al. (2001). A Case for Economy Grid Architecture for Service-Oriented Grid Computing. Proceedings of the International Parallel and Distributed Processing Symposium:10th IEEE International Heterogeneous Computing Workshop (HCW 2001), San Francisco, California, USA, IEEE CS Press.

Buyya, R., H. Stockinger, et al. (2001). Economic Models for Management of Resources in Peer-to-Peer and Grid Computing. Proceedings of International Conference on Commercial Applications for High-Performance Computing, Denver, Colorado, USA, SPIE Press.

Computing, N. t. G., http://www-106.ibm.com/developerworks/grid/newto/.

Folding@Home Scientific Background, http://www.stanford.edu/group/pandegroup/folding/science.html.

Foster, I. and A. Iamnitchi "On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing." Avaialble at:http://people.cs.uchicago.edu/~anda/papers/foster_grid_vs_p2p.pdf.

Foster, I. and C. Kesselman (1997). "Globus: A Metacomputing Infrastructure Toolkit." International Journal of Supercomputer Applications **11**(2): 115-128.

Foster, I. and C. Kesselman, Eds. (1999). The Grid: Blueprint for a Future Computing Infrastructure. USA, Morgan Kaufmann.

Grimshaw, A. and W. Wulf (1997). "The Legion Vision of a Worldwide Virtual Computer." Communications of the ACM **40**(1).

Litzkow, M., M. Livny, et al. (1988). Condor - A Hunter of Idle Workstations. Proceedings of the 8th International Conference of Distributed Computing Systems (ICDCS 1988), San Jose, CA, IEEE CS Press, USA.

Medvinsky, G. and C. Neuman (1993). NetCash: A design for practical electronic currency on the Internet. Proceedings of 1st the ACM Conference on Computer and Communication Security.

Napster, http://www.napster.com/.

Neuman, C. and G. Medvinsky (1995). Requirements for Network Payment: The NetCheque Perspective, San Francisco, USA.

Nisan, N., S. London, et al. (1998). Globally Distributed computation over the Internet: The POPCORN project. International Conference on Distributed Computing Systems (ICDCS98), Amsterdam, The Netherlands, IEEE CS Press, USA.

Oram, A. (2001). Peer-to-Peer: Harnessing the Power of Disruptive Technologies. USA, O'Reilly Press.

Paypal, http://www.paypal.com.

SETI@Home, http://setiathome.ssl.berkeley.edu/.

Shirky, C. (2000). What is P2P. And What Isn't, http://www.openp2p.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html.

TechWeb, http://www.techweb.com.

Waldspurger, C., T. Hogg, et al. (1992). "Spawn: A Distributed Computational Economy." IEEE Transactions on Software Engineering **18**(2): 103-117.

WWW: A Brief History of the Internet, http://cse.stanford.edu/class/sophomore-college/projects-01/distributed-computing/html/body_history.html.