Compute Power Market: Towards a Market-Oriented Grid

Rajkumar Buyya
School of Computer Science
Monash University, Melbourne,
VIC 3145, Australia
rajkumar@csse.monash.edu.au

Sudharshan Vazhkudai
Department of Computer and Science
University of Mississippi, University,
MS 38677, USA
chucha@john.cs.olemiss.edu

CPM Portal: `http://www.computepower.com`

Abstract

The Compute Power Market (CPM) is a market-based resource management and job scheduling system for grid computing on Internet-wide computational resources, particularly low-end personal computing devices. It transforms the metacomputing environment into a computational market wherein one can solve problems by renting computational power, storage, and special services from idle resources (computers). The CPM primarily comprises of markets, resource consumers, resource providers and their interactions. It supports various economic models (commodity market model, contract-net/ tendering, and auction) for resource pricing and mapping between service consumers and providers. This paper proposes a decentralized computation market with multiple markets and numerous consumers and providers spread across the grid environment. The paper further discusses the basic architecture and the components involved in markets, consumers and providers namely, a Market Server, a Market Resource Agent, a Market Resource Broker and a Market Trader and scheduler used for negotiation and job deployment.

Keywords: *Grids, Computational Economy, Markets, Internet Computing*

1. Introduction

The recent technological advances in high-performance networking and computing, coupled with their availability as commodity components, have revolutionized the way we do computing. The trend in high-performance computing is to move away from proprietary supercomputers to those based on commodity hardware and software components. This has led to the popularity of clusters of computers, interconnected through local/system-area networks, as a platform for solving large-scale compute intensive problems. Today, the Internet/Web has become pervasive and millions of computers and users are online. Most of the time, of these users are browsing the Web, carrying out word processing tasks, or reading emails that consume less than 25% of computing power. Also, when machines are idle, they are mostly running screen savers. This unused and idle computational power available on machines across the Internet can be utilized for solving resource intensive applications.

A number of projects such as SETI@Home [16] and distributed.net [21] have successfully exploited this paradigm for solving specific application areas. They have adopted custom design and system architecture for computing on volunteer resources. Recently, several commercial ventures have begun to extend this concept a step further for business advantage [4] and they include ProcessTree [12], Popular Power [13], Mojo Nation [14], United Devices [15], Entropia [11], and Parabon [10].

Although using volunteer computers' idle CPU cycles for solving supercomputing problems appears simple, realizing a flexible and widely acceptable resource management, scheduling, re-programmable machinery, and general-purpose paradigm for application programming is a complex task. This is mainly due to resources' geographic distribution, heterogeneity, distributed ownership with different policies and priorities, varying loads, reliability, and availability conditions. Another key issue that these systems need to address is a regulation of resource demand and supply for creating a computational marketplace, which is missing in most of these systems software infrastructure. We propose a market-based economic paradigm for resource management that helps in addressing all of these issues in a simplified manner, since economic institution

has been proven to be the best mechanism for regulating demand and supply. Furthermore, it offers incentives for volunteers to share their computational resources and encourages consumers to optimally utilize resources by balancing timeframe and access-costs. Even the profit can be shared with a market for coordinating users. The Scientific American also highlights the importance of computational economy in metacompututing and suggested that without it, metacomputing may arrive with whimper, not a bang [3]. In this paper we propose a Compute Power Market (CPM) system that adopts economic paradigm for resource management and scheduling of computations across Internet-wide volunteer resources.

In the rest of the paper, we discuss a brief comparison with related systems and present CPM system architecture and its components comprising of a set of decentralized Markets, Market Information Services, Market Resource Agents and Brokers. We present the design issues to be considered during the implementation and then conclude with future work.

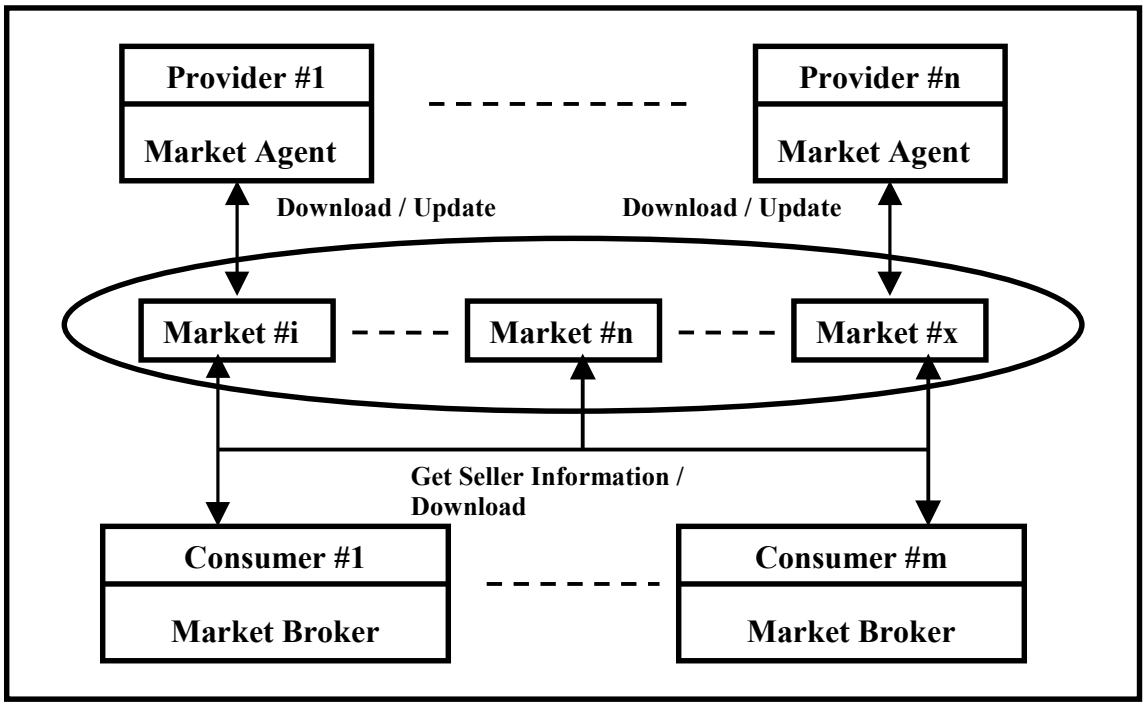

Figure 1: The Compute Power Market

2. Related Work

Using idle computers to perform useful processing is not a new concept. The idea came into the limelight largely due to the success of the SETI@Home project, which distributed huge scientific datasets, collected from observatories, to millions of home computers in order to perform computations on them. Recently, a number of commercial ventures have originated that extend this concept a step further. Examples of such endeavors are ProcessTree, Popular Power, Mojo Nation, United Devices, Entropia, and Parabon. These systems allow the home computer owner to specify the kind of research for which they wish to allocate their computers. Yet another category is the metacomputing systems such as Globus [5] and Legion [6] that concentrate on high-end resources such as supercomputers and clusters managed using queuing systems and offer infrastructures for distributing high-end application loads on them. None of these, including commercial grid systems adopt economic paradigm for resource management and scheduling. In [2], we presented Grid Architecture for Computational Economy (GRACE) for high-end Grid computing systems and we believe that it can be adopted for low-end machines for global computing with suitable changes in implementation architecture and the underlying infrastructure. The key changes will be in terms of replacing those middleware services by CPM

services targeted for personal computing devices or machines.

A few other systems such as Java Market [17], Popcorn [20], and JaWS [19] build market-oriented environments to harness the processing power of a small network-of-computers configuration or Web-based systems. Our approach blends the basic idea behind these attempts with computational economy principles to build a true market-oriented Internet-scale computational Grid (software CPU).

3. CPM Architecture

3.1. Overview

The Compute Power Market is primarily composed of three entities, namely: a Market, a Resource consumer and a Resource provider. The market works in the following simple manner: consumers and providers announce their desire to buy or sell compute power from the market. As part of expressing their desire to contribute to the market or to benefit from the market, they register with the market. We will look at the details involved while discussing the structure of the market.

When resource providers register with the market, they obtain/download a "***Market Resource Agent***" (MRA) from the market and deploy on their resource; consumers obtain a "***Market Resource Broker***" (MRB) from the market. These agents help synchronize and maintain the flow of interaction amongst the three entities. The intent of the Market Resource Agent is to update the Market with the latest information about the resource provider and to accept, deploy and launch the job; the intent of the Market Resource Broker is to help the consumer find an appropriate provider based on the information provided by the Market (Figure 1). The resource information provided by Market agents is maintained in the CPM database for providing Market Information Services (MIS).

Various economic models (such as commodity market, contract net/tenders, auctions) need to be supported for resource trading and establishing prices in the CPM grid marketplace. It should also be noted that both resource providers and consumers would prefer to maximize their own objectives, i.e., consumers would like to execute their applications within minimum cost/budget and providers would like to increase their profit (by charging high or attracting rich consumers like in the real marketplace). We also need to have a provision for the Market to charge its users for serving as a mediator between them, i.e., some percentage of resource provider's benefit or consumer's price-quote can be credited to the market for maintaining its business like in real exchanges (stock market).

Let us look at these entities more closely by discussing their design and architecture.

3.2. The Market

One can perceive the market to be a passive agent in the CPM, in that it acts as a mediator between consumers and providers, by providing the following services:

- Repository of information on providers
- Agents for consumers and providers
- Mechanisms for updating the information
- Interaction with other markets

The CPM can comprise of a number of markets supporting their own consumers and providers. This facilitates decentralization of control and adds to the stability of the CPM. Markets can communicate and interact among themselves to synchronize information. Let us now look at the various components of a market.

A market in the CPM consists of the following components (Figure 2):

1. A Market Entry Index
2. Provider Domain
3. Consumer Domain
4. Market Control Unit
5. Communication Unit

3.2.1. Market Entry Index

The Market Entry Index is a repository that consists of information about providers and consumers within the market's domain. Each provider within the domain of Market #i would typically have an entry (if they registered with the market) in the Market Entry Index. This entry is a record comprising of information supplied by the Market Resource Agent in that particular provider. The Market Entry Index is updated whenever resource providers change their preferences and is used when a decision has to be made while matching consumer requirements with provider capabilities.

The Market Entry Index also contains information about other markets. This information includes details such as: addresses of markets, capabilities, etc. Thus, this information

can be treated as a link to other markets.

The Market Entry Index could easily be a database holding information about all providers that could be queried upon.

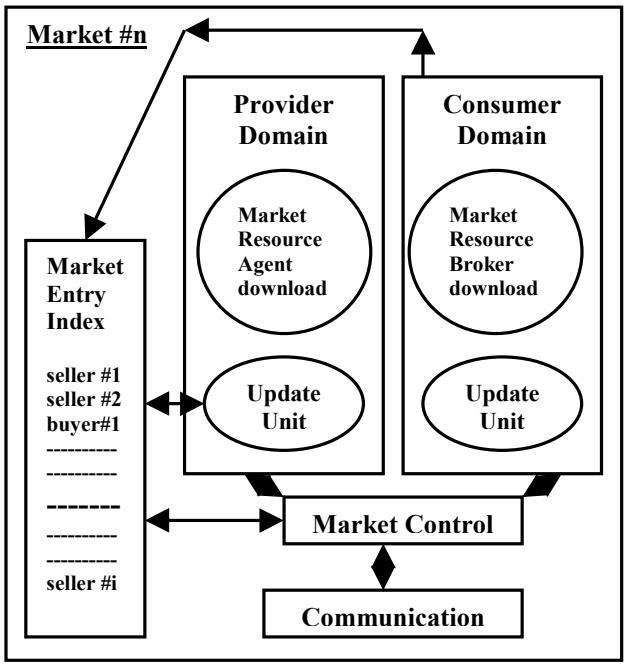

Figure 2: Market Architecture

3.2.2. Provider Domain

The provider domain, as the name suggests, is concerned with resource providers in the CPM. The provider domain primarily comprises of:

1. Market Resource Agent Download Unit

The *Market Resource Agent Download Unit* is responsible for keeping track of the download of market resource agents by providers. Providers contact the market to register and download the market resource agent, a program using which they let the market know the status of their resources, preferences, or pricing rules. The download unit presents a simple information sheet, which providers complete before they download the agent. Upon completion of the download, the unit initializes an entry in the market entry index for that particular provider. It is also responsible for tracking duplicates, i.e., limiting one market resource agent per host, etc.

2. Update Unit

The update unit is concerned with updating the entry corresponding to a particular provider. Whenever the provider specification changes, the Market Resource Agent sends information about its resource to the market. This information is gathered by the update unit, which updates the particular provider's entry with the latest information. The update unit is also responsible for de-registering a resource provider from the market.

3.2.3. Consumer Domain

The consumer domain is similar to the provider domain but is concerned with consumers in the market. It, similar to the provider domain, comprises of a "**Market Resource Broker Download Unit**" and an "**Update/Query Unit**". These two units perform the actions of monitoring downloads and updating information on consumers, similar to their counterparts in the provider domain. The one thing that is slightly different in the update/query unit is that, it does not have to update the market entry index with information about consumers, but instead has to query it about provider details whenever such a request arrives. The querying process could sometimes lead to searching through multiple markets in order to obtain provider information. It can achieve this by following the link (to other markets) in the market entry index.

3.2.4. Market Control Unit

The Market Control Unit is the brain of each market in the CPM. It controls the behavior of the market by:

1. Channelizing/Regulating requests

The market control unit acts as a conduit for directing requests to particular domains in the market. For example, it redirects download requests to relevant agents and update requests to relevant update units. Requests can be differentiated based on their ids.

2. Monitoring the market behavior

The control unit is further concerned with monitoring the behavior of the market. For example, it could monitor the kinds of requests from consumers and providers, and over a period of time be able to predict the kinds of requests that arrive at the market. This information can be used as an indication for prospective consumers and providers. Consumers and providers can buy and sell from/to markets that have a demand (market tendencies) for requests that match their preferences.

It is further concerned with monitoring the timely servicing of each request. If any one of the market units is temporarily dysfunctional, it dispatches relevant messages and generates necessary timeouts and further attempts to restart failed units, thereby contributing to the fault tolerance of the market.

3. Synchronizing with other markets

The control unit is also responsible for synchronizing with other markets; maintaining information and pointers to other markets and their tendencies. This is useful while having to redirect requests to relevant markets, etc.

3.3. The Market Resource Agent

A potential provider, after deciding to contribute his resource to the CPM, contacts the market to download a Market Resource Agent. The agent primarily comprises of the following components (Figure 3):

1. A GUI Front End
2. A Backend

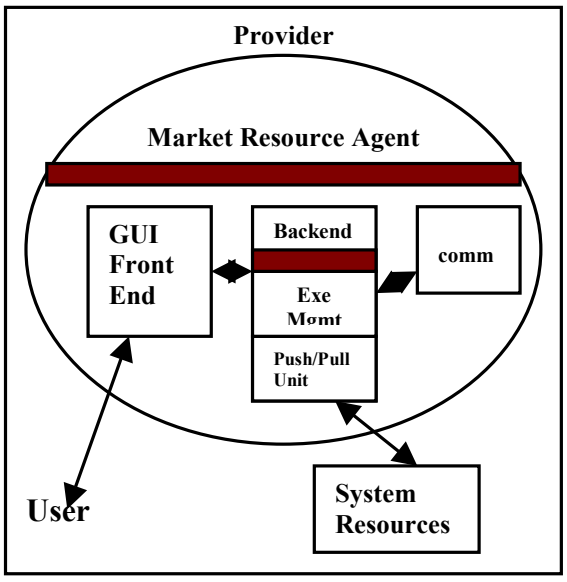

Figure 3: Market Agent Architecture

3.3.1. GUI Front End – Active Screensaver

The GUI front end provided as part of the Market Resource Agent is responsible for providing an interface for the user, facilitating the user to provide details on his resource. As part of the details, the user specifies the policies under which his resource might be used, system configuration, pricing details, and various other details.

Yet another functionality of the front end is to provide a screen saver utility, which is activated during the resource's idle time. This screen saver, through the back end, acts as an intermediary that communicates with the resource consumer for executable deployment and launching.

3.3.2. Back End

The backend primarily comprises of Push, Pull units and an Executable Management Unit. The Exe Mgmt unit handles the necessary details involved in consumer executable deployment and launching. The Pull unit is concerned with extracting dynamic information from a resource, for example: available memory, idle time, number of processes, etc; while the Push unit is concerned with flushing this data to the Market using the communication unit.

3.3.1. The Market Resource Broker

The consumer uses Market Resource Broker (MRB) (Figure 4) services for interacting with CPM grid. The *resource broker* acts as a mediator between the user application and CPM resources. It is responsible for the management of the whole experiment on the CPM grid. We would like to provide resource brokers for the following types of applications:

- Sequential applications (for both Java-based and legacy applications).
- Parallel Applications including tightly coupled master-worker type applications.
- Parameter Sweep applications (by providing tools like Nimrod/G [1]).

Broker Functions

When the user submits these applications with their requirements to a suitable resource broker, it performs the following:

1. Resource Discovery
2. Matching job requirements against provider capabilities
3. Perform trading between matched entities depending on suitable economic model for establishing service access cost.
4. Select resources that fit user requirements
5. Match jobs to resources
6. Deploy jobs on resources
7. Monitor and Steer computations
8. Perform load profiling for future usage

9. Perform rescheduling, if required.
10. When job execution is finished, gather results back to the user home machine.
11. Record all resource consumption details for payment processing purpose.
12. Perform cleanup and post-processing, if required.

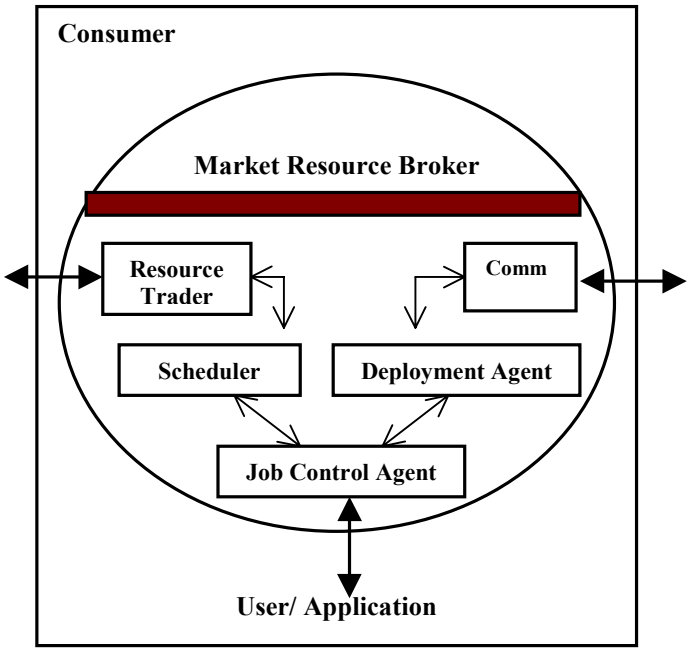

Figure 4: Market Broker Architecture

Broker Components

The resource broker is made up of following components:

- Job control agent
- Market explorer
- Resource Trader
- Scheduler
- Deployment Agent

The *job control agent* is a persistent central component responsible for shepherding a job through the system by interacting with all other components of the broker. It accepts application requirements of the form: job requires x amount of memory, runs for y duration, etc. These requirements are specified in terms of attributes. A client for each class of application can help formulate user requirements and communicate them to the job control agent.

The components that are specifically responsible for managing economy of computations in CPM Grid are the *schedule adviser*, *trade manager*, and *trader server*. The schedule adviser uses services of *grid explorer* for resource discovery (using the Market information services), trade server for negotiating access costs from trader server, and scheduling algorithms for identifying mappings (jobs to resources) that meet user requirements (deadline and cost minimization). The trade server decides access costs based on resource-owner defined charging algorithms/policies and interacts with accounting system for recording usage details and billing the user as per negotiation.

4. Design Issues

In this section, we will discuss a few key design issues involved in the realization of the architecture explained. These design issues are primarily a few basic questions regarding the CPM:

1) Is the Market an LDAP server?

The most vital question concerns the protocol involved in the exposure of resource (provider) capabilities, i.e., the mechanism with which the Market server would publish provider details. Systems such as Globus use the Lightweight Directory Access Protocol (LDAP) [8], for Grid Information Services [7]. LDAP is primarily designed to be used in services that require lesser updates and could prove a performance bottleneck [9]. Moreover, in our case, having to use LDAP implies that each and every consumer and provider is forced to install LDAP servers.

For these reasons, we use a traditional database implementation, wherein the Market Server maintains a database for consumer and provider details. Application requirements and provider details are represented as fields in database entries. Extracting information on either consumer or provider details results in the formulation of typical database queries. We are currently tending towards Java and JDBC for the Market Server implementation.

2) How will the various CPM components communicate?

Through out our discussion on CPM, we have elaborated little on the communication amongst its various components. Communication amongst the various CPM components is achieved using the *Comm Unit*. A few scenarios of the necessary handshaking between these components are:

a) The Market Resource Agent having to

update the Market Server about the provider details, whenever there is a change in the provider specification.

b) Market Resource Broker, at the consumer end, requiring to contact the Market Server in order to obtain potential providers.

c) Consumer and provider communicating among themselves to deploy the executable.

d) Market-to-Market interaction.

Mentioned above are amongst the few basic, mandatory, handshaking involved in the CPM grid system. Given the flexibility, scalability and proliferation of Java, these tasks can be achieved using Java sockets, remote method invocation, and Java networking packages.

3) Will the CPM be programmable?

The eventual success of any system rests on its ability to be programmed and altered according to user definitions. This requires the definition and specification of programmable APIs. We intend to provide a set of APIs for the Market Server, the Market Resource Agent and the Broker so that these components can be made flexible enough.

4) What types of programs will the CPM support?

An ideal environment would support the execution of all possible executables. Our initial prototype concentrates on Java based programs to primarily prove the concept behind such a large scale Internet computing platform. However, trusted legacy applications can be executed in the CPM environment.

5) What kind of information do resource providers and resource consumers advertise and how will they be mapped?

Yet another seemingly trivial task is that of information discovery and mapping. Listed below are a few issues:

a) A potential provider can advertise details such as: machine architecture, software availability, usage policies, cost considerations, etc. The Market resource agent should discover such details and various others employing a mix of dynamic and static strategies.

b) A consumer application that requires a resource, can submit its requirements such as: program requirements, memory, disk, amount of time the resource is required, cost willing to pay, etc.

c) The Market resource broker performs a mapping of these requirements against optimal capabilities, attempting to obtain a suitable mapping. Potential candidates for performing such a mapping are XML or the Classified Advertisement matchmaking tool [18].

These are only a few of the vital, lingering questions. As we proceed ahead with the implementation specifics, we would be able to comment further on our design decisions and choices.

5. Security in CPM

In this section, we highlight a few security-related issues in the CPM:

1. In the CPM, clients contribute their resources voluntarily with an implicit trust. In such a scenario, utmost care should be taken to ensure client safety and security, i.e., care should be taken to ensure that the programs executed at the client end do not, in any way, harm them. One way of accomplishing such security is by executing trusted code (code from parties that the CPM can properly authenticate). Another alternative is to execute programs within secure sandbox emulation.
2. The highly diverse nature of the CPM implies programs executed will also tend to be varied with various levels of confidentiality. In such a scenario, maintaining the propriety of source programs that are executed in client sites becomes very relevant. A potential problem arises when mischievous clients attempt to decompile and reverse-engineer the logic behind these programs. An initial step against such security breach is incorporating obfuscated code - logic in the programs is purposefully convoluted to discourage attacks (Examples: Entropia and Parabon).
3. Yet another aspect is that of sabotage tolerance, a typical problem faced by highly dynamic systems such as: Seti@Home, wherein millions of client computers perform calculations on datasets and return their results to a central site. There is no guarantee that these results are correct and are not sabotaged. Seti@Home handles this problem by dispatching each dataset to at least two client sites. Moreover, the only thing that is probed in Seti@Home is a particular type of signal, which can be easily

verified given their client base. In a system like CPM, where diverse applications can be executed, there is not an easy method of verifying results. In [22], work is being done in areas of sabotage tolerance and result verification.

Above mentioned are only but a few vital issues with regards to building a complex environment such as, CPM.

6. Conclusion

With the proliferation of the Internet, efficient techniques to harness the processing power of millions of computers, spread across diverse administrative domains and geographic distances, have emerged. Various approaches have been initiated to achieve this goal – community initiative such as SETI@Home, metacomputing initiatives such as Globus, Legion, GRACE, and commercial ventures such as: Entropia and Parabon. In this paper, we present a market-oriented grid environment that applies economic initiatives to Internet computing, thereby presenting a motivating factor for computer owners to contribute their resources. To this end, we have designed a market-based architecture where consumers and providers can buy and sell computing power based on an underlying economic architecture. Initiatives are underway to realize the architecture described in this paper.

Acknowledgements

We would like to thank CPM project members David Sanchez and Alvaro Suarez (Departamento de Ingeniería Telemática, Spain) for their design for implementing Market Resource Agent and Christopher Nebergall for his thoughts on CPM security.

References

[1] Buyya, R., Abramson, D., and Giddy, J., *Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid*, HPC ASIA 2000, China, IEEE CS Press, USA, 2000.

[2] Rajkumar Buyya, Jonathan Giddy, David Abramson, *An Economy Grid Architecture for Service-Oriented Grid Computing*, 10th IEEE International Heterogeneous Computing Workshop (HCW 2001), In conjunction with IPDPS 2001, San Francisco, USA, April 2001.

[3] Gibbs W., Cyber View—World Wide Widgets, Scientific American, San Francisco, USA-http://www.sciam.com/0597issue/0597cyber.html

[4] R. Buyya, Grid Computing Infoware (Info Centre) - http://www.gridcomputing.com/

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

[6] S. Chapin, J. Karpovich, A. Grimshaw, *The Legion Resource Management System*, Proceedings of the 5th Workshop on Job Scheduling Strategies for Parallel Processing, April 1999.

[7] Fitzgerald S., Foster I., Kesselman C., Laszewski G.V., Smith W., and Tuecke S., *A Directory Service for Configuring High-Performance Distributed Computations*, Proceedings of the 6th IEEE Symposium on High Performance Distributed Computing, pp. 365-375, 1997.

[8] Howes T., and Smith M., *LDAP: Programming Directory Enabled Applications with Lightweight Directory Access Protocol*, Macmillan Technical Publishing, 1997.

[9] Smith W., Waheed A., Meyers D., and Yan J., *An Evaluation of Alternative Designs for a Grid Information Service*, Proceedings of the 9th IEEE Symposium on High Performance Distributed Computing, pp. 185-192, 2000.

[10] Parabon — http://www.parabon.com

[11] Entropia Inc. - http://www.entropia.com/

[12] ProcessTree–http://www.processtree.com/, Distributed Science Inc, Nov. 2000.

[13] Popular Power - http://www.PopularPower.com/

[14] Mojo Nation - http://www.mojonation.net/

[15] United Devices - http://www.uniteddevices.com/

[16] SETI@Home— http://setiathome.ssl.berkeley.edu/

[17] Amir Y., Awerbuch B., Borgstorm B.S., *The Java Market: Transforming the Internet into a Metacomputer*, Technical Report CNDS-98-1, Johns Hopkins University, 1998.

[18] Raman R., Livny M., Solomon M., *Matchmaking: Distributed resource management for high throughput computing*, Proceedings of the 7th IEEE Symposium on High Performance Distributed Computing, 1998.

[19] Spyros Lalis and Alexandros Karipidis, *An Open Market-Based Framework for Distributed Computing over the Internet*, First IEEE/ACM International Workshop on Grid Computing (GRID 2000), Dec. 2000, Bangalore, India: Springer Verlag, Germany.

[20] Noam Nisan, Shmulik London, Ori Regev, Noam Camiel, *Globally Distributed computation over the Internet - The POPCORN project*, International Conference on Distributed Computing Systems (ICDCS'98) 1998. Also a poster in WWW6 - Sixth International World Wide Web Conference, Santa-Clara, 1997.

[21] Distributed.net — http://www.distributed.net/

[22] Bayanihan — http://www.cag.lcs.mit.edu/bayanihan/