InteGrade: Object-Oriented Grid Middleware Leveraging Idle Computing Power of Desktop Machines*

Andrei Goldchleger† Fabio Kon
Alfredo Goldman Marcelo Finger
Department of Computer Science
University of São Paulo
{andgold,kon,gold,mfinger}@ime.usp.br
http://gsd.ime.usp.br/integrade

May 8, 2003

Abstract

Grid computing technology improves the computing experiences at organizations by effectively integrating distributed computing resources. However, just a small fraction of currently available Grid infrastructures focuses on reutilization of existing commodity computing resources. This paper introduces *InteGrade*, a novel object-oriented middleware Grid infrastructure that focuses on leveraging the idle computing power of shared desktop machines. Its features include support for a broad range of parallel applications and mechanisms to assure that the owners of shared resources do not perceive any loss in the quality of service. A prototype implementation is under construction and the current version is available for download.

1 Introduction

In recent years we have witnessed an impressive growth in the use of computers in various fields of research, including physics, chemistry, biology, and economics. Even corporations rely on the intensive use of computing power to solve problems such as financial market simulations and studies for accurate oil well drilling. The movie industry makes intensive use of computers to render movies using an increasing number of special effects.

The recent introduction of clusters of commodity computers brought down the costs of the hardware needed to perform intensive computations. However, this solution has major drawbacks. First, traditional clusters are composed of dedicated machines. This means that when no computation is being carried out, machines remain idle, normally inaccessible to other users. Second, the whole cluster infrastructure demands a lot of physical space, a temperature controlled environment, and measures to deal with the noise produced by cluster nodes. These problems may seem negligible when we consider a 16 node cluster, but when the cluster grows in size, they have to be considered carefully.

Meanwhile, when we consider the computing resources available at corporations and universities, we see that they typically have hundreds or thousands of desktop machines, which are used by workers as their personal workstations or by students in instructional and research laboratories. When analyzing the usage of each of these machines we typically conclude that they sit idle for a significant amount of time. Even when the computer is in use, it normally has a large portion of idle resources. When we consider the night period, we conclude that most of the times they are not used at all. This situation lives in contradiction with the huge demand for computational resources already described. The need for and waste of resources often coexist in the same institution. The problem is that organizations lack a software infrastructure to allow the efficient use of these idle resources.

This paper introduces *InteGrade*, a novel Grid middleware architecture to solve the contradiction mentioned above. The architecture enables a wide range of parallel applications to execute in a distributed environment, benefiting from the power of the hardware already available in organizations. This is achieved by integrating user desktop machines and machines in shared laboratories in a intranet or wide-area Computational Grid [FK99].

*This work is supported by a grant from CNPq, Brazil, process # 55.2028/02-9.
†Andrei Goldchleger is partially supported by a grant from CAPES, Brazil

Differently from other grid approaches, InteGrade is based on state-of-the-art middleware technology for distributed objects, namely, the CORBA [OMG02] industry standard for distributed object systems. This allows us to leverage existing services [OMG98], such as Naming, Trading, Transactions, and Persistence, shortening development and maintenance time. InteGrade services are exported as CORBA IDL interfaces being accessible from almost any programming language and operating system.

An important requirement for InteGrade is that users who decide to share their machines with the Grid shall not perceive any drop in the quality of service provided by their applications. InteGrade's architecture was carefully designed with this requirement in mind. Thus, client machines use a very lightweight CORBA implementation and the access to its hardware resources is carefully controlled by a user-level scheduler.

Although many applications can benefit from this environment, it is clear that parallel applications will benefit the most. Usually they have to be executed on dedicated resources, such as Beowulf clusters [Beo03], but InteGrade allows them to execute over many shared resources, bringing the power of parallel computing to institutions that cannot afford a multi-node dedicated cluster. Dedicated resources can also be part of InteGrade grids, either remaining dedicated or converted to workstations and shared in the grid.

In such a dynamic environment it is difficult to schedule the execution of applications, since an idle resource may become busy again without further notice. To minimize this problem, InteGrade's architecture includes a component for collecting and analyzing usage patterns, a mechanism that based on usage information and statistics, can determine the probabilities of an idle node to become busy again. When tuned properly, this mechanism can help schedulers to forecast if an idle machine will stay idle for a significant amount of time or if it is going to be busy again in a few seconds.

The InteGrade project is a multi-university, 3-year initiative that started in August 2002. Section 2 discusses InteGrade's related work and Section 3 explains the major requirements and challenges the project will face. In Section 4 we present the proposed architecture and in Section 5 we describe which parts of the architecture have already been implemented, which parts are still under development, and on what technologies the implementation is based. Finally, we present our conclusions in Section 6.

2 Related Work

In recent years, many Grid related projects surfaced, enabling new ways of resource sharing and integration.

The Globus Project [Glo03, FK97] provides an infrastructure for the integration of several computers, spanning different geographical locations into a single grid system. It provides a toolkit that allows the construction of grid-enabled applications in an incremental fashion. While Globus focuses on building the software infrastructure for a broad range of machines, including high-end ones, InteGrade focuses on leveraging the idle processing power from commodity workstations, taking appropriate measures to ensure that their users do not feel any drop in the quality of service. Another difference is that InteGrade is being built using the CORBA industry standard, which facilitates the interaction between the middleware system and applications through the use of well defined IDL interfaces. The use of CORBA also allows the middleware platform to be based on a distributed object model leveraging existing CORBA services, such as Trading, Naming, and others.

Legion [Leg03, GW96] provides a middleware infrastructure that enables applications to benefit from execution in a distributed and parallel environment. It provides its own specific runtime library and builds its services on top of a unified object model [LG96]. InteGrade differs from Legion in its use of CORBA instead of a proprietary distributed object model. InteGrade also has a deeper focus on idle resource management and commodity hardware.

Condor [Con03] may be considered the pioneer [LLM88] of Grid systems. As InteGrade does, it focuses on harvesting idle computing power from workstations in order to perform useful computation, such as *High Throughput Computing* [LBRT97] tasks. It provides scheduling and monitoring to applications without the need of modifying them. It also provides checkpointing but this requires the application to be re-linked with a specific library. However, support for parallel applications is currently quite limited, since some computers in the system should be configured [Wri01] as partially-reserved nodes, like nodes on a Beowulf cluster. The reservation might not be feasible, for example, if the node is used by an employee. Checkpointing of parallel applications is currently under development, and can address this issue. Differently from Condor, InteGrade is being built with parallel applications in mind from the beginning. We also plan on featuring mechanisms for collecting and analyzing usage patterns, which can be used to forecast the behavior of grid nodes to improve scheduling. Finally, InteGrade uses CORBA as its communication protocol rather than a specific one used in Condor.

The SETI@home project [SET03] built an infrastructure to solve a single problem, SETI (*Search for Extraterrestrial Intelligence*), and obtained support from thousands of users, leveraging the power of

hundreds of thousands of commodity workstations throughout the world. It is accredited as the problem that has received the largest amount of computing time in history. Despite its tremendous success, this project has several limitations when compared to InteGrade, the most notable being the impossibility of solving different problems. Other limitations include the lack of support for parallel applications that demands communication between computing nodes, the necessary intervention of the client machines to specify when the application can run and the impossibility of using resources of a *partially* idle node.

BOINC (*Berkeley Open Infrastructure for Network Computing*) [Boi03] is considered SETI@home's successor. It introduces many features that were missing in SETI@home, such as the possibility of using the grid to solve different problems, limited support for communication between parallel application nodes and checkpointing. Although development is in its beginning, the features planned for the full version lack usage pattern collection and analysis, one of InteGrade features which will help to choose where applications should run. Differently from InteGrade, BOINC lacks general support for parallel applications, being more suitable for applications with negligible data dependencies between its nodes.

3 Architectural Requirements

One of InteGrade's most important requirements is to leverage idle resources of commodity computers. That requisite demands that the middleware have the means for determining the activity status in each of the grid nodes. To address that requisite, we need an information service that gathers all relevant information from each node. We also need the definition of what should be considered a candidate node to run a grid application. One could argue that a candidate node is any node that has any amount of free resources at a certain moment. This could be the basic definition used by the system, but to obtain the collaboration of a large number of users, we need to empower users with the means to determine when their machine resources will be exported to the grid and what portion of its resources can be used by grid applications. Thus, the system must provide a flexible and user-friendly way of letting resource providers share their machines as they want. On the other hand, we must also keep in mind that the vast majority of resource providers will not be knowledgeable users, so the system must provide sensible default values for its parameters to protect providers from degradation in the quality of service.

Another important requirement related to leveraging idle resources is ensuring that an application does make progress in its execution. We need the means to ensure that application execution evolves even in a dynamic environment in which nodes can turn from idle to busy without further notice. One solution is to use a checkpointing mechanism to ensure progress by periodically saving the application execution state. Since the grid can encompass different platforms and operating systems, this checkpointing must be machine and operating system independent to permit migration of computation across grid nodes. In order to minimize checkpointing situations, we make use of usage pattern collection and analysis. This procedure is based on information gathered by resource managers. Node usage information for short time intervals (e.g., 5 minutes) is grouped in larger intervals called periods. After that, the system shall apply clustering algorithms [JW83] to this data in order to extract behavioral categories. It is expected that these categories will map to common usage periods such as lunch-breaks, nights, holidays, working periods, and so on. From that mapping it will be possible to predict the time-span in which a machine will be idle. This is an evolutionary process: as data is being collected and analyzed new categories can appear, others can disappear.

Another major requirement is to provide support for a wide range of parallel applications, which is not the case in existing grid systems. Most grid initiatives provide support for parallel applications with little or no communication among application nodes. Some parallel applications demand more in terms of inter-process communication than others. The different demands in terms of communication suggest that different applications should be scheduled to different computers, respecting the differences regarding the network connectivity associated to each grid node. So, the distributed resource management service must also provide information on the kind of network connection available in each part of the grid. That kind of information will help schedulers to create virtual topologies based on the needs of parallel applications. With this kind of support, a grid user may, for example, submit the following request to InteGrade: *execute application $\mathcal{X}$ in two groups of 50 nodes, each group connected internally by a 100 Mbps network and the two groups connected by a 10 Mbps network; each node should have at least 16 MB of RAM and a CPU of at least 500 MIPS.*

The need to ensure application progress is even more challenging when parallel applications are considered. Measures that are adequate when sequential programs are considered are not directly applicable in parallel execution environments. Consider checkpointing, for example. If the system has to checkpoint a parallel application, what should it do with ongoing communications? How to determine that a process migrated to another machine, thus requiring all pending communications to be redirected? This kind of issue can render parallel checkpointing prohibitive, due to large overheads. Usage pattern analysis plays an important role since the scheduler can place parallel applications on idle nodes with lower probability

of becoming busy before the computation is completed. Although usage patterns attenuate the problem, it does not solve it entirely since it can only provide a *hint* of what is going to happen and cannot guarantee the future availability of resources. We still need a model that saves the state of computation periodically, providing milestones that can be used to resume the application in case of crashes or when there is need for migration. To ensure that, InteGrade adopts BSP [Val90] as the model for parallel computation; imposing frequent synchronizations among application nodes.

Finally, security should be considered through the whole architecture. The most important requirement is to ensure that users who decide to export its resources to the grid do not have its personal files and overall private information exposed or damaged in any way. To ensure that, we are investigating the use of Java and general sandboxing [GWTB96] to protect from malicious code execution; authentication, and cryptography.

4 Architecture

InteGrade grids are structured in clusters, each consisting of groups from one to approximately one hundred computers, which can be shared workstations or machines dedicated to the grid. Clusters are then arranged in a hierarchy, allowing a single InteGrade grid to encompass millions of machines. The hierarchy can be arranged in any convenient manner. Figure 1 depicts the major kinds of components in a InteGrade cluster. The *Cluster Manager* node represents one or more nodes that are responsible for managing that cluster and communicating with managers in other clusters. A *User Node* is one belonging to a grid user who submits applications to the cluster. A *Resource Provider Node*, typically a workstation, is one that exports part of its resources making them available to the grid users. A *Dedicated Node* is one reserved for grid computation. This kind of node is shown to stress that, if desired, InteGrade can also encompass dedicated resources. Note that those categories can overlap; for example, a node can be a User Node and a Resource Provider node at the same time.

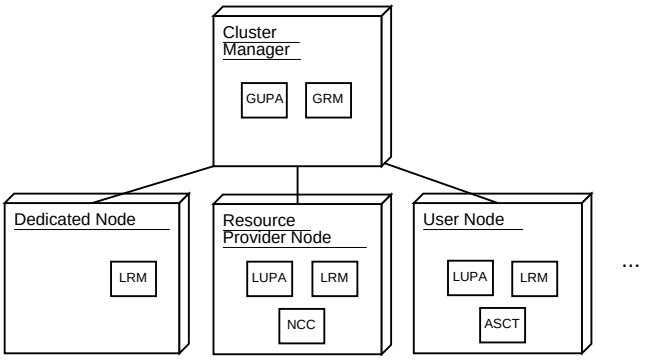

Figure 1: InteGrade's Intra-Cluster Architecture

The *Local Resource Manager* (**LRM**) and the *Global Resource Manager* (**GRM**) cooperatively handle intra-cluster resource management. The LRM is executed in each cluster node, collecting information about the node status, such as memory, CPU, disk, and network usage. LRMs send this information periodically to the GRM, which uses it for scheduling within the cluster. This process is called the *Information Update Protocol*.

The GRM and LRMs also collaborate in the *Resource Reservation and Execution Protocol*, which works as follows. When a user submits an application for execution, the GRM selects an candidate node for execution, based on resource availability and application requirements. For that the GRM uses its local information about the cluster state as a hint for locating the best nodes to execute an application. After that, the GRM engages in a direct negotiation with the selected nodes to ensure that they actually have the sufficient resources to execute the application at that moment and, if possible, reserves the resources in the target nodes. In case the resources are not available in a certain node, the GRM selects another candidate node and repeats the process. The information, execution and reservation protocols are based on previous work in the 2K Resource Management Subsystem [KYH+01]. A recent extension of this protocol [MK02] implemented by our group allows the GRM to engage in information updates, resource negotiation, and reservation across a collection of clusters organized in a wide-area hierarchy. Figure 2 depicts a possible cluster hierarchy. Resources in any of the clusters can be transparently acessed if needed.

Similar to the LRM/GRM cooperation, the *Local Usage Pattern Analyzer* (**LUPA**) and the *Global Usage Pattern Analyzer* (**GUPA**) handle intra-cluster usage pattern collection and analysis. The LUPA

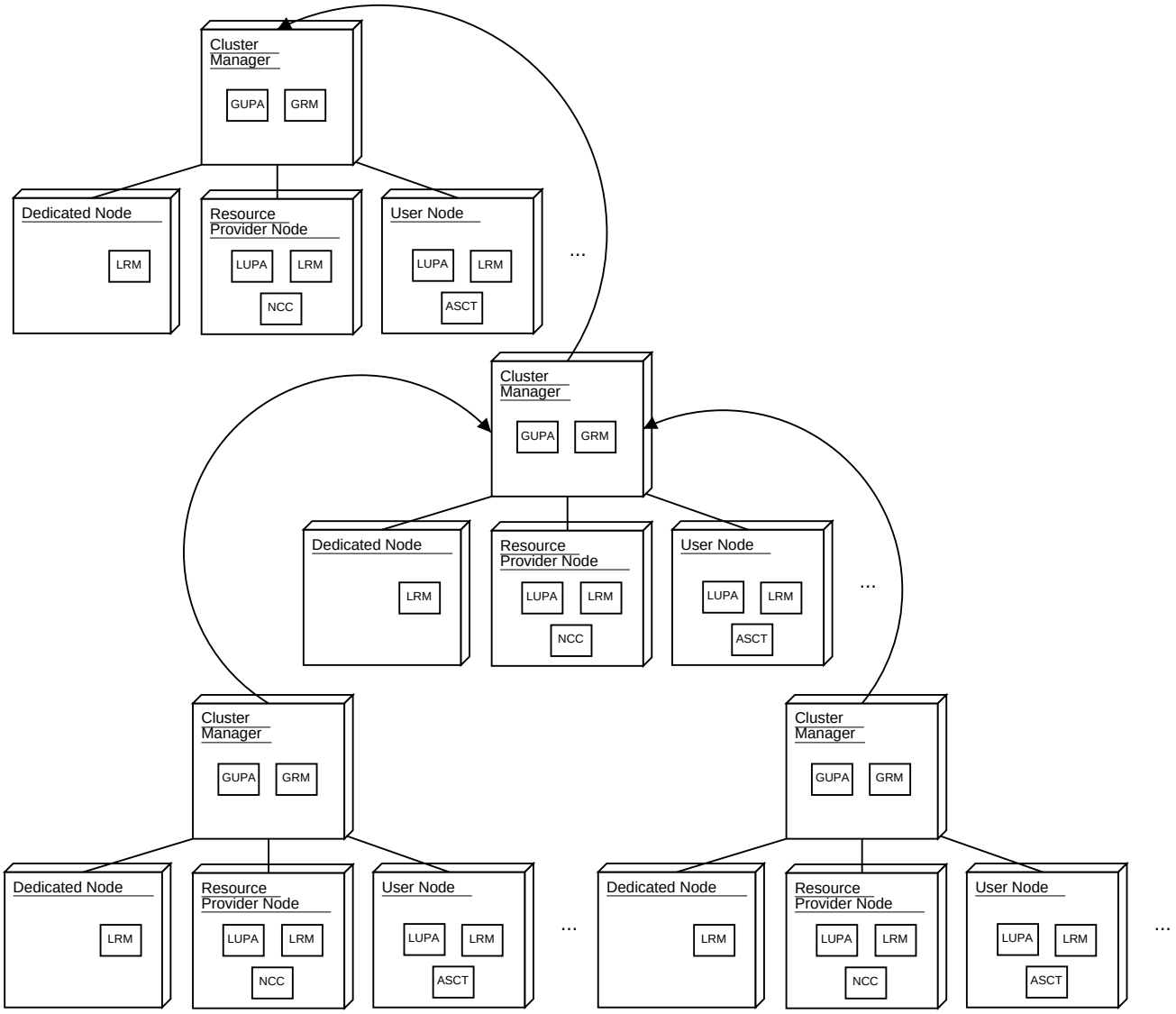

Figure 2: InteGrade's Inter-Cluster Architecture

executes in each cluster node that is a user workstation [1] and collects data about its user usage patterns. Based on long series of data, it derives usage patterns for that node throughout the week. Each node's usage pattern is periodically uploaded to the GUPA. This information is made available to the GRM, which can make better scheduling decisions due to the possibility of predicting a node's idle periods based on its usage patterns.

The *Node Control Center* **(NCC)** allows the owners of resource providing machines to set the conditions for resource sharing, if they so wish. Parameters such as periods in which they do not want their resources to be shared, the portion of resources that can be used by grid applications (e.g., 30% of the CPU and 50% of its physical memory), or definitions as to when to consider their machine idle can be set using this tool. The *Dynamic Soft Real Time Scheduler*, part of DSRT [NhCN98], enforces the conditions of the resource owners.

The *Application Submission and Control Tool* **(ASCT)** allows InteGrade users to submit applications for execution in the grid. The user can specify execution prerequisites, such as hardware and software platforms, resource requirements such as minimum memory requirements, and preferences, like rather executing on a faster CPU than on a slower one. The user can also use the tool to monitor application progress.

5 Implementation Status and Ongoing Work

Although we had implemented the intra- and inter-cluster protocols for information updates, resource reservation, and execution in the 2K system, we are currently re-implementing these protocols from scratch in the new InteGrade middleware platform. This was necessary due to InteGrade's strict requirements with respect to ensuring that the quality of service perceived by users sharing their machines with the grid would not be affected.

The new middleware platform is based on UIC-CORBA [RKC01], a very small memory footprint CORBA-compatible implementation (90 KB for a Client/Server ORB). In this new middleware platform, we have implemented the intra-cluster information protocol that allows LRMs to send node status to GRMs. The LRM is currently implemented in C++ using UIC-CORBA. The GRM, which runs on a server node, is implemented in Java on top of JacORB [Jac03]. The GRM uses the JacORB Trader to store the information it receives from the LRMs. We are also investigating the use of other small-footprint ORBs such as LORB [Cer03] and Orbix/E [Orb03].

Ongoing work includes the implementation of the intra-cluster execution protocol, that will allow applications to be remotely executed in an InteGrade cluster. We also started to collect information about node's usage in order to develop node usage patterns that will be used on LUPA and GUPA. We expect to have a first working version of InteGrade by the end of the first semester of 2003. Source-code and documentation of the latest version is available at the InteGrade Web site.

6 Conclusion

InteGrade will provide a middleware infrastructure to enable applications to leverage the idle computing power from commodity computers. Its key features are support for a broad range of parallel applications, use of advanced object-oriented techniques on architectural design and development, and node usage pattern collection, analysis and prediction. This infrastructure will unlock the power of distributed parallel computing in organizations that cannot afford to have dedicated resources. It has also a great potential for lowering the level of waste of computational resources in today's computing infrastructure.

References

[Beo03] Beowulf. `http://www.beowulf.org`, 2003.

[Boi03] Boinc. `http://boinc.berkeley.edu`, 2003.

[Cer03] Renato Cerqueira. LORB: A small footprint CORBA compliant ORB. Personal Communication, 2003.

[Con03] Condor. `http://www.cs.wisc.edu/condor`, 2003.

[FK97] Ian Foster and Carl Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputing Applications*, 2(11):115–128, 1997.

[FK99] Ian Foster and Carl Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, San Francisco, 1999.

[1] The LUPA is not executed in dedicated nodes that can only be used remotely.

[Glo03] Globus. `http://www.globus.org`, 2003.

[GW96] Andrew S. Grimshaw and Wm. A. Wulf. Legion – A View From 50,000 Feet. In *Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing*, Los Alamitos, California, August 1996. IEEE Computer Society Press.

[GWTB96] I. Goldberg, D. Wagner, R. Thomas, and E. A. Brewer. A Secure Environment for Untrusted Helper Applications: Confining the Wiley Hacker. In *Proceedings of the USENIX Security Symposium*, July 1996.

[Jac03] JacORB. `http://www.jacorb.org`, 2003.

[JW83] Richard A. Johnson and Dean Wichern. *Applied Multivariate Statistical Analysis*. Prentice-Hall, 1983.

[KYH+01] Fabio Kon, Tomonori Yamane, Christopher Hess, Roy Campbell, and M. Dennis Mickunas. Dynamic Resource Management and Automatic Configuration of Distributed Component Systems. In *Proceedings of the 6th USENIX Conference on Object-Oriented Technologies and Systems (COOTS'2001)*, San Antonio, Texas, February 2001.

[LBRT97] Miron Livny, Jim Basney, Rajesh Raman, and Todd Tannenbaum. Mechanisms for High Throughput Computing. *SPEEDUP Journal*, 11(1), June 1997.

[Leg03] Legion . `http://www.cs.virginia.edu/~legion`. 2003.

[LG96] Michael J. Lewis and Andrew Grimshaw. The Core Legion Object Model. In *Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing*, Los Alamitos, California, August 1996. IEEE Computer Society Press.

[LLM88] Michael Litzkow, Miron Livny, and Matt Mutka. Condor - A Hunter of Idle Workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems*, pages 104–111, June 1988.

[MK02] Jeferson Roberto Marques and Fabio Kon. Distributed Resource Management in Large-Scale Systems *(in Portuguese)*. In *Proceedings of the 20th Brazilian Symposium on Computer Networks*, pages 800–813, Búzios, Brazil, May 2002.

[NhCN98] Klara Nahrstedt, Hao hua Chu, and Srinivas Narayan. QoS-aware Resource Management for Distributed Multimedia Applications. *Journal of High-Speed Networking, Special Issue on Multimedia Networking*, 7:227–255, 1998.

[OMG98] OMG. *CORBAservices: Common Object Services Specification*. Object Management Group, Framingham, MA, December 1998. OMG Document 98-12-09.

[OMG02] OMG. *CORBA v3.0 Specification*. Object Management Group, Needham, MA, July 2002. OMG Document 02-06-33.

[Orb03] Orbix/E. `http://www.iona.com/products/orbix-e.htm`, 2003.

[RKC01] Manuel Román, Fabio Kon, and Roy Campbell. Reflective Middleware: From Your Desk to Your Hand. *IEEE Distributed Systems Online*, 2(5), July 2001. Available at `http://dsonline.computer.org/0105/features/rom0105_print.htm`.

[SET03] SETI@home. `http://setiathome.ssl.berkeley.edu`, 2003.

[Val90] Leslie G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33:103–111, 1990.

[Wri01] Derek Wright. Cheap cycles from the desktop to the dedicated cluster: combining opportunistic and dedicated scheduling with Condor. In *Proceedings of the Linux Clusters: The HPC Revolution conference*, Champaign - Urbana, IL, June 2001.