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

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 computers. 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 wide range of parallel applications and mechanisms to assure that 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. 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. Even 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.

Meanwhile, when considering the computing resources available at corporations and universities, one sees 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 one concludes that they sit idle for a significant amount of time. This situation lives in contradiction with the huge demand for computational resources in certain areas. 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 resources in shared laboratories in a intranet or wide-area Computational Grid [1].

Differently from other grid approaches, InteGrade is based on state-of-the-art middleware technology, namely, the CORBA industry standard for distributed object systems. This allows us to leverage existing services, shortening development and maintenance time. InteGrade's architecture was carefully designed to prevent users sharing resources to perceive any drop in the quality of service provided by its applications. 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, 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 foresee 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.

2 Related Work. The Globus Project [2] provides an infrastructure for the integration of computers spanning different geographical locations into a single grid system. Differently from Globus, InteGrade focuses on leveraging the idle processing power of 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 has a completely object-oriented architecture built using the CORBA industry standard, leveraging existing CORBA services and facilitating communication in heterogeneous environments.

Legion [3] provides a middleware infrastructure that enables applications to benefit from execution in a distributed and parallel environment. 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 [4] may be considered the pioneer 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* tasks. Differently from Condor, InteGrade is being built, from the beginning, with parallel applications in mind. InteGrade features 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 proprietary one used in Condor.

The SETI@home project `setiathome.ssl.berkeley.edu` built an infrastructure to solve a single problem, SETI (*Search for Extraterrestrial Intelligence*). Despite its tremendous success, this project has a limited scope: the most notable limitation is 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*, `boinc.berkeley.edu`), SETI@home's successor, 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. Differently from InteGrade, BOINC lacks general support for parallel applications, being more suitable for applications with negligible data dependencies between its nodes.

3 Architecture. InteGrade grids are structured in clusters, each consisting of groups from one to approximately one hundred computers. Clusters are then arranged in a hierarchy, allowing a single InteGrade grid to encompass potentially millions of machines. Figure 1 depicts the major types of components in a InteGrade cluster. The *Cluster Manager* 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 grid. A *Resource Provider Node* is one that exports part of its resources, making them available to grid users. A *Dedicated Node* is one reserved for grid computation. Note that these categories may overlap: for example, a node can be a User Node and a Resource Provider at the same time.

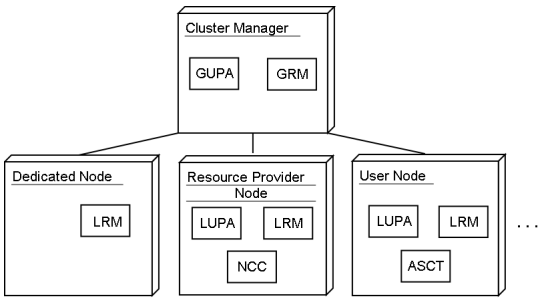

Figure 1: InteGrade's Intra-Cluster Architecture

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 node status, such as memory, CPU, disk, and network utilization. LRMs send this infor-

mation 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 grid user submits an application for execution, the GRM selects candidate nodes 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 [5]. A recent extension of this protocol [6] implemented by our group allows the GRM to engage in information updates, resource negotiation, and reservation across a collection of clusters connected through the Internet.

Similarly 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 executes in each cluster node that is a user workstation 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, or definitions as to when to consider their machine idle can be set using this tool. The *Application Submission and Control Tool* **(ASCT)** allows InteGrade users to submit grid applications for execution. The user can specify execution prerequisites, resource requirements, and preferences. The user can also use the tool to monitor application progress.

4 Implementation Status and Ongoing Work. We have already implemented the intra-cluster information protocol. The LRM is currently implemented in C++ using UIC-CORBA, a very small memory footprint CORBA-compatible ORB. The GRM, which runs on a server node, is implemented in Java on top of JacORB. 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. Ongoing work includes the implementation of the intra-cluster execution protocol and collecting and analyzing information about usage patterns. We expect to have a first working version of InteGrade by the end of the first semester of 2003.

5 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 wide 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

[1] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, 1999.

[2] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *Int. Journal of Supercomputing Applications*, 2(11):115–128, 1997.

[3] A. Grimshaw and W. Wulf. Legion, A View From 50,000 Feet. In *Proc. 5th IEEE HPDC*, August 1996.

[4] M. Litzkow, M. Livny, and M. Mutka. Condor - A Hunter of Idle Workstations. In *Proc. 8th IEEE ICDCS*, pages 104–111, June 1988.

[5] Fabio Kon et al. Dynamic Resource Management and Automatic Configuration of Distributed Component Systems. In *Proc. 6th USENIX COOTS*, 2001.

[6] Jeferson Marques and Fabio Kon. Distributed Resource Management in Large-Scale Systems. In *Proc. 20th Brazilian Symposium on Computer Networks*, pages 800–813, Búzios, Brazil, May 2002.