Sneha-Samuham: An Instant Messenger based Grid Computing Model

N.V. Palankeswara Rao, D. Janaki Ram, A. Vijay Srinivas
Distributed and Object Systems Group
Department of Computer Science and Engineering
Indian Institute of Technology Madras, Chennai - 600 036
India
{palan, djram, avs}@cs.iitm.ernet.in

Abstract

More often, there is a considerable amount of unused computing power available over the Internet. Collection of these computing resources with adequate software support can form computational grids over the Internet, which can be used for solving computation intensive applications. This paper presents an instant messenger kind of interface to aggregate the geographically distributed resources into a computational grid over the Internet. It also describes a model for parallel execution of an application over such a computational grid. Two case studies have been implemented over the proposed model: a parallel solution to the numerical integration problem and a neutron shielding simulation.

1 Introduction

Internet has become an attractive means to share information across the globe. The idea of "Grid Computing" has emerged from the fact that the Internet can also be used for several other purposes such as sharing the computing power, storage space and other scientific devices etc. The term "Grid" is chosen as an analogy to the electric power grid where it provides consistent, pervasive and ubiquitous power irrespective of its source. A Grid can be seen as a seamless, integrated computational and collaborative environment.

An instant messenger kind of interface is much useful to share the computing resources over the Internet. Using this kind of simple interface, we can do away the complex resource publication and location protocols to publish and locate the resources. Resources can be requested and shared as simple as that of making friends over the Internet using "yahoo" kind of instant messenger. Since this kind of sharing will occur among the known and trusted users, it does not require strong security and authentication policies also. A simple public and private key kind of authentication mechanism may be sufficient. Apart from the other difficulties of grid computing such as heterogeneity, fault tolerance, security etc., there should be mechanisms to exploit the global computing power in the application programs as well as in the run-time environment. Most of the grid computing models do not support the auto splitting of a task, even though it is possible for many classes of the applications. These systems expect the user to submit a batch of jobs to the system and the scheduler schedules these jobs on the participating nodes.

We have proposed a grid computation model called Sneha-Samuham[1], which contains an instant messenger kind of interface for sharing computational resources and a model for parallel computation of coarse-grained parallel applications over the Internet. The model can also provide the auto splitting of a task for certain classes of applications. Users can harness the computing power of "friendly" computers over the Internet for parallel processing. Sneha-Samuham model handles heterogeneity in hardware architectures and operating systems as well as communication latencies and varying processor speeds.

Rest of the paper is organized as follows. Next section gives an overview about various components and functionalities of the model. Section 3 presents case studies and their performance characteristics. Section 4 reviews some existing global computing models and finally section 5 concludes the work and gives some future directions.

[1]In Sanskrit, "Sneha-Samuham" means "a group of friends".

2 Sneha-Samuham: An Instant Messenger based Parallel Computing Grid Model

In Sneha-Samuham model, a computer over the Internet can donate its computing power to other computer as well as it can use the computing power of other computers. The one which donates its computing power is called a "donor" and the one which accepts is called an "acceptor". An acceptor can use the computing power of donors for executing coarse-grained parallel applications. Figure 1 gives an abstract view of the model. The system contains three major components: run-time environment, user interface, and the functional library support. Run-time environment is responsible for getting the sub-tasks from the user interface, migration and execution of these sub-tasks on the remote machines over the Internet and getting the results back to the node, where the computation is initiated. For performing these tasks it uses the functions provided in the functional library support.

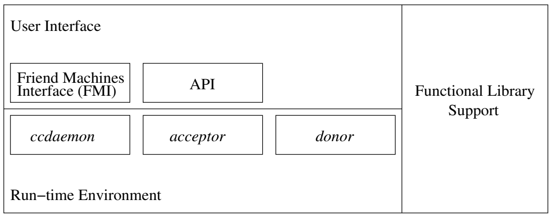

Figure 1: An abstract view of the model

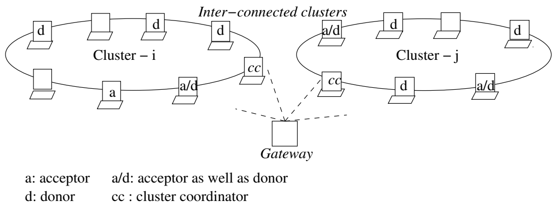

Figure 2: System Topology

The Internet can be treated as an interconnection of individual LANs (Local Area Networks) as shown in the figure 2. We call all the nodes within a LAN as a *cluster*. A designated node in each *cluster* will act as a mediator between the nodes inside its *cluster* and the outside world. The mediator is called as a *Cluster Coordinator* (CC). So each *cluster* contains a CC. Any node inside a cluster (including CC) can act either as a "donor" or as an "acceptor" or both as an "acceptor" as well as a "donor" or may not participate in the grid computing at all.

Our user interface is the first to provide an "instant messenger" kind of interface for machines participating in the parallel computation. We call this interface as the "Friend Machines Interface (FMI)". The FMI provides the following functionality.

- Owner of a cluster can request owners of other clusters over the Internet to become a friend cluster to his/her cluster. If he/she accepts, then both the clusters become friends to each other. Each Cluster Coordinator (CC) includes the name of the other CC in its friend clusters list.
- One can accept or reject the request made by others over the Internet to make his cluster friend to them. Protocol similar to that of "yahoo messenger" for discovering chat friends can be used.
- It displays all friend CCs of a CC and highlights the live friend CCs. Along with the friend CCs, other information such as number of donors available in each friend cluster, their Grid Computation Capacity Factors (GCCFs, defined later in this paper) and communication latencies can also be displayed when the user wants.
- User or the application can select any number of donors from friend clusters for parallel computing.

Here it is assumed that each cluster is owned by a user or a group of users. It is a realistic assumption as each LAN is owned by a researcher group or an organization. They decide whether their cluster will participate in the grid computing or not. They can make any node as a CC, acceptor or a donor by running the corresponding softwares on them. The owner(s) can make their cluster to friend of any other clusters by using the FMI described above. A cluster including all its friend clusters forms a *computational grid*. A number of such grids can exist over the Internet. Apart from the friend machines interface, the user interface also contains the Application Programming Interfaces (APIs) to submit parallel applications to the grid. Depending on the application type (or class), user has to use the interface provided in the API part of the user interface to submit a parallel task to the grid. For different class of applications, different APIs have to be used. The functionality included in that API actually contains the logic to split a parallel task into sub-tasks. This logic is different for different classes of applications. So for each class of applications a different API is provided in the user interface, using which the user can submit his parallel task to the grid. However implementing these kind of APIs is not possible for all applications as some class of applications are inherently not parallelizable.

The run-time environment of the system contains three components that run as daemons on the respective nodes: *acceptor*, *donor* and *ccdaemon*. A *donor* daemon runs on each donor node and an *acceptor* daemon runs on each acceptor node. A node which acts as both donor as well as acceptor runs both the daemons. Every CC runs a daemon called *ccdaemon* to coordinate between the acceptors and the donors. The node where the parallel task is submitted should have the *acceptor* daemon running and the nodes where *donor* daemon is running can participate in the parallel computation by sharing work from acceptors.

2.1 Computation Model

The computation model follows the *Master-Worker*[1] processes model. After getting the task, the acceptor contacts its local *ccdaemon* and gets the addresses and Grid Computation Capacity Factors (GCCFs) of requested number (or available number) of friend machines, and spawns a master process for the assigned application. The master process executes the functionality provided in the API, specified for that class of applications. The master process splits the task into sub-tasks of appropriate granularity depending on the number of available friend machines and their current computing capabilities. After that, it migrates each sub-task to the corresponding node. The *donor* daemon that receives the sub-task to be computed, spawns a worker process for that sub-task and returns the results after the execution is over. The division of task among the available lightly loaded nodes is done as follows. Computational capability of a node over the grid can be measured from the factor called Grid Computation Capacity Factor (GCCF), $f = \frac{sm}{lt}$, where s, m and l are processor speed, memory and average load on a node and t is the communication latency from a node, from where it received the job to be executed.

If there are n nodes with GCCFs, $f_1, f_2, f_3, ..., f_n$.

Let,

$F = f_1 + f_2 + f_3 + ... + f_n$

And if the total task size is G, then

Grain size assigned to the first node, $g_1 = (\frac{f_1}{F})G$

Grain size assigned to the second node, $g_2 = (\frac{f_2}{F})G$

.

.

Grain size assigned to the n_{th} node, $g_n = (\frac{f_n}{F})G$

The above task distribution is applicable only to data parallel applications. This technique may have to be extended to apply for task parallel applications.

The current model is implemented using C programming language. UNIX network system library is used for network related services. Any node running UNIX based operating systems such as Linux, Solaris and HP-UX can participate in the Sneha-Samuham computational grid. Heterogeneity in hardware architecture is taken care by using eXternal Data Representation (XDR) for all communications among the participating nodes. Full design and implementation details can be found in [2].

3 Case Study

We used all identical machines (PII, 267 MHz processors), running Linux operating system for our case studies. We have chosen two applications which fall into the same class and hence can use the same API in the user interface for splitting and submitting the tasks to the grid. However due to space limitation, here we are presenting only one case study called, neutron shielding simulation [3]. In this application, when a neutron beam strikes a lead sheet perpendicularly, a simulation has to be performed to find out the number of neutrons that can penetrate from the other side of the lead sheet. The amount of computing power required is directly proportional to the number of neutrons that hit the lead sheet. We call this number of neutrons as problem size of that application.

Figure 3 plots the speedups achieved for various number of machines inside a single cluster for problem sizes 10^7 and 10^8. In the first case(10^7), the speedup saturates early because the computation to communication ratio is less compared to that of the second case(10^8). There was no extra load on the machines, when these tasks were executing. Our model can use the high GCCF machines from friend clusters when local machines are having low GCCF, which is an attractive grid computing feature. An experiment has been conducted to show the gain in speedup due to usage of no load machines from a neighboring cluster instead of using partially loaded machines from the local cluster. Figure 4 shows the difference in speedup when half of the participating machines are 50% loaded in the local cluster (Single Cluster) and using the no load machines from a neighboring friend cluster instead of the 50% loaded machines from the local cluster (Multiple Clusters) for problem size of 10^8.

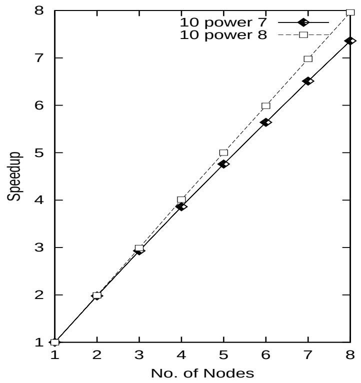

Figure 3: Neutron Shielding Simulation - Single Cluster

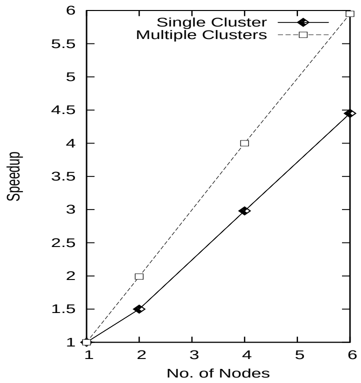

Figure 4: Neutron Shielding Simulation - Multiple Clusters

4 Related Work

Javelin[4], ATLAS[5] and Popcorn[6] are java based batch processing grid computing models. These models follow the broker kind of architectures where the user submits a batch of jobs to the intermediate nodes and these nodes intern schedule the jobs at available nodes participating in the grid computing. These models expect the user to split the task into sub-tasks and submit them to the system. Also since these models are Java based, performance may become a cause for concern.

Globus[7] provides low level services such as resource location, resource management, communication and security, on which higher level metacomputing software can be built. Instead of competing, our model complements the Globus features. As Globus provides standard protocols for resource location, resource management, communication, and security at lower layers, the proposed parallel computing model can be built on top of these layers. Like Globus, Legion[8] also provides the grid computing middleware services, but it is built based on object oriented principles.

Condor-G[9] and Nimrod/G[10] are application level schedulers for parameter sweep applications. These two are extensions of the famous cluster computing models Condor[11] and Nimrod[12] respectively, and built on top of the Globus middleware services. Condor-G tries to schedule the tasks near to the data required by a task. Nimrod/G tries to meet the soft real-time deadlines at the same time it tries to keep the cost of computation as low as possible.

5 Conclusions

This is the first attempt made to provide an "instant messenger" kind of friend machines interface to form computational grids over the Internet. Due to hierarchical management of the nodes, the model is scalable. The present user interface is minimal and it supports only a single class of parallel applications. More APIs can be included in the user interface to support larger categories of parallel applications. We believe that by using this kind of simple interface such as APIs, users can use the system with less effort for their computation intensive applications. The present Sneha-Samuham model emphasizes more on parallel processing of grid applications. It has not addressed the other issues in grid computing such as resource location, fault tolerance, security etc. Some of these functionalities can be provided by using the toolkits provided by the Globus. The work is under progress to provide these extensions.

References

[1] Gregory R. Andrews, "Paradigms for Process Interaction in Distributed Programs", *ACM Computing Surveys*, Vol. 23, No. 1, March 1991, pp. 49-90.

[2] N.V. Palankeswara Rao, D. Janaki Ram and A. Vijay Srinivas, "Sneha-Samuham: A Parallel Computing Model over Grids", *Technical Report*, No. IITM-CSE-DOS-2002-15.

[3] Ward Cheney and David Kincaid, "Numerical Mathematics and Computing", Fourth Edition, Brooks/Cole Publishing Company, 1999.

[4] M.O. Neary, B.O.Christiansen, P. Capello and K.E.Schauser, "Javelin: Parallel Computing on the Internet", *Future Generation Computer Systems*, Vol. 15, October 1999, pp 659-674.

[5] J. Baldeschwieler, R. Blumofe, E. Brewer, "ATLAS: An Infrastructure for Global Computing", *Seventh ACM SIGOPS European Workshop on System Support for Worldwide Applications*, 1996.

[6] N. Camiel, S. London, N. Nisan, and O. Regev. "The POPCORN Project: Distributed Computation over the Internet in Java", *6th International World Wide Web Conference*, April. 1997.

[7] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit", *International Journal of Super Computer Applications and High Performance Computing*, Vol. 11, No. 2, Summer 1997, pp. 115-128.

[8] A. Grimshaw and W. Wulf, "The Legion Vision of a World-wide Virtual Computer", *Communications of the ACM*, Vol. 40, No. 1, January 1997, pp. 39-45.

[9] James Frey, Todd Tannenbaum, Ian Foster, Miron Livny, and Steven Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids", *Journal of Cluster Computing*, volume 5, pages 237-246, 2002.

[10] R. Buyya, D. Abramson, and J. Giddy, "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, 2000.

[11] Litzkow, M. J., M. Livny, and M. W. Mutka, "Condor- A Hunter of Idle Workstations", Proceedings of *IEEE International Conference on Distributed Computing Systems*, IEEE Press, Piscataway, N. J. , 1988, pp. 104-111.

[12] Abramson D., Sosic R., Giddy J. and Hall B., "Nimrod: A Tool for Performing Parametised Simulations using Distributed Workstations", *The 4th IEEE Symposium on High Performance Distributed Computing*, Virginia, August 1995.