Tunghai Science Vol. 5: 107−124
July, 2003

Construct a Grid Computing Environment on Multiple Linux PC Clusters

Chuan-Lin Lai and Chao-Tung Yang [*]

Abstract

Internet computing and Grid technologies promise to change the way we tackle complex problems. They will enable large-scale aggregation and sharing of computational, data and other resources across institutional boundaries. And harnessing these new technologies effectively will transform scientific disciplines ranging from high-energy physics to the life sciences. In this paper, a grid computing environment is proposed and constructed on multiple Linux PC Clusters by using Globus Toolkit (GT) and SUN Grid Engine (SGE). The experimental results are also conducted by using the pi problem, prime problem, and matrix multiplication to demonstrate the performance.

Keywords: Cluster Computing, Grid Computing, Globus ToolKit, PC Clusters, SUN Grid Engine.

1. Introduction

Grid computing, most simply stated, is distributed computing taken to the next evolutionary level. The goal is to create the illusion of a simple yet large and powerful self managing virtual computer out of a large collection of connected heterogeneous systems sharing various combinations of resources. The standardization of communications between heterogeneous systems created the Internet explosion. The emerging standardization for sharing resources, along with the availability of higher bandwidth, are driving a possibly equally large evolutionary step in grid computing [1, 14, 15].

The Infrastructure of grid is a form of networking. Unlike conventional networks that focus on communication among devices, grid computing harnesses unused processing cycles of all computers in a network for solving problems too intensive for any stand-alone machine. A well-known grid computing project is the SETI (Search for Extraterrestrial Intelligence) @Home project [30], in which PC users worldwide donate unused processor cycles to help the search for signs of extraterrestrial life by analyzing signals coming from outer space. The project relies on

[*] Department of Computer Science and Information Engineering, Tunghai University, Taichung 407, TAIWAN

individual users to volunteer to allow the project to harness the unused processing power of the user's computer. This method saves the project both money and resources.

Another key technology in the development of grid networks is the set of middleware applications that allows resources to communicate across organizations using a wide variety of hardware and operating systems. The Globus Toolkit [2] is a set of tools useful for building a grid. Its strength is a good security model, with a provision for hierarchically collecting data about the grid, as well as the basic facilities for implementing a simple, yet world-spanning grid.

Globus will grow over time through the work of many organizations that are extending its capabilities. More information about Globus can be obtained at http://www.globus.org. The accepted standard in this application space is the Globus Toolkit. The Globus Toolkit is a middleware product designed to facilitate grid computing, and also like Linux is available under an “open source” licensing agreement for free use.

The promise of grid computing is to provide vast computing resources for computing problems like the SETI example that require supercomputer type resources in a more affordable way. Grid computing also offers interesting opportunities for firms to tackle tough computing tasks like financial modeling without incurring high costs for super computing resources. The developers of the Globus Toolkit envision that grid computing will become the pervasive paradigm for providing computing recourses to large collaborative projects and virtual organizations.

The organization of this paper is as follow. In section 2, we make a background review of Cluster Computing, MetaComputing and Grid Computing. In section 3, it is our hardware and software configuration. In section 4, grid computing environment is proposed and constructed on multiple Linux PC Clusters by using Globus Toolkit (GT) and SUN Grid Engine (SGE). The experimental results are also conducted by using the pi problem, prime problem, and matrix multiplication to demonstrate the performance. The experimental result are presented and discussed. We conclude this study in section 4. The software installation and setup detail are presented in Appendix.

2. Background Review

2.1 Cluster Computing

The first cluster computing was a NASA effort called Beowulf. Beowulf was started in 1994 and the first effort consisted of a 16-node cluster made up of commodity off the shelf

(COTS) systems interconnected with Ethernet. While this approach does not try to exploit the excess computing power in the network, the use of COTS computers and standard network architectures means that Beowulf class systems are inexpensive to build and operate and can offer supercomputer levels of processing power.

Scalable computing clusters, ranging from a cluster of (homogeneous or heterogeneous) PCs or workstations to SMP (Symmetric MultiProcessors), are rapidly becoming the standard platforms for high-performance and large-scale computing. A cluster is a group of independent computer systems and thus forms a loosely coupled multiprocessor system. A network is used to provide inter-processor communications. Applications that are distributed across the processors of the cluster use either message passing or network shared memory for communication. A cluster computing system is a compromise between a massively parallel processing system and a distributed system. An MPP (Massively Parallel Processors) system node typically cannot serve as a standalone computer; a cluster node usually contains its own disk and equipped with a complete operating systems, and therefore, it also can handle interactive jobs. In a distributed system, each node can function only as an individual resource while a cluster system presents itself as a single system to the user.

Since a Beowulf cluster is a parallel computer system, it suits applications that can be partitioned into tasks, which can then be executed concurrently by a number of processors. These applications range from high-end, floating-point intensive scientific and engineering problems to commercial data-intensive tasks. Uses of these applications include ocean and climate modeling for prediction of temperature and precipitation, seismic analysis for oil exploration, aerodynamic simulation for motor and aircraft design, and molecular modeling for biomedical research [10, 11, 21, 22].

The previous study [11] lists four benefits that can be achieved with clustering. These can also be thought of as objectives or design requirements:

- **Absolute scalability:** It is possible to create large clusters that far surpass the power of even the largest standalone machines. A cluster can have dozens of machines, each of which is a multiprocessor.
- **Incremental Scalability:** A cluster is configured in such a way that it is possible to add new systems to the cluster in small increments. Thus, a user can start out with a modest system and expand it as needs grow, without having to go through a major upgrade in which an existing small system is replaced with a larger system.
- **High availability:** Because each node in a cluster is a standalone computer, the failure of one node does not mean loss of service. In many products, fault tolerance is handled automatically in software.

- **Superior price/performance:** By using commodity building blocks, it is possible to put together a cluster with equal or greater computing power than a single large machine, at much lower cost.

2.2 MetaComputing and Grid Computing

The term “MetaComputing” was coined around 1987 by NCSA Director, Larry Smarr [12]. But the genesis of metacomputing at NCSA took place years earlier, when the center was founded in 1986. Smarr's goal was to provide the research community with a “Seamless Web” linking the user interface on the workstation and supercomputers. With the advent of networking technologies such as Ethernet and ATM. it has become possible to connect computers for the widespread, efficient sharing of data. As high performance local- and wide-area networks have become less expensive, and as the price of commodity computers has dropped, it is now possible to connect a number of relatively cheap computers with a high-speed interconnect, to affect a local distributed computing cluster.

Clusters of distributed computers, as well as high performance serial and parallel machines can be interconnected, speaking common communications protocols, to form a large virtual supercomputer. The general trend for computational capacity has thus been to move from monolithic single-processor computers in the early 1970's, to multi-processor parallel computers, in which the interprocessor bandwidth was high and the latency quite low, to clusters of commodity workstations with comparatively high bandwidth and latency, and ultimately to so-called MetaComputing environments, which connect heterogeneous clusters of high-end and low-end computers to form a virtual supercomputer.

The term metacomputer was coined to describe a collection of possibly heterogeneous computational nodes which can be treated as a single virtual computer for both resource management and remote execution purposes. General metacomputing environments allow users to submit serial or parallel programs and have tasks or jobs run on the virtual computer. An alternative definition of metacomputing is provided by Gehring and Reinefeld: “a monolithic computational resource provided by software that allows the transparent use of a network of heterogeneous, distributed computers” [12, 13].

The Globus project [2] provides a new infrastructure to metacomputing. The globus make the metacomputing standardized and normalized. To take the metacomputing into the Grid Computing.

Grid computing (or the use of a computational grid) is applying the resources of many computers in a network to a single problem at the same time - usually to a scientific or technical problem that requires a great number of computer processing cycles or access to large amounts of data. A well-known example of grid computing in the public domain is the ongoing SETI (Search for Extraterrestrial Intelligence) @Home project [30] in which thousands of people are sharing the unused processor cycles of their PCs in the vast search for signs of "rational" signals from outer space. According to John Patrick, IBM's vice-president for Internet strategies, "the next big thing will be grid computing."

Grid computing requires the use of software that can divide and farm out pieces of a program to as many as several thousand computers. Grid computing can be thought of as distributed and large-scale cluster computing and as a form of network-distributed parallel processing. It can be confined to the network of computer workstations within a corporation or it can be a public collaboration (in which case it is also sometimes known as a form of peer-to-peer computing).

A number of corporations, professional groups, university consortiums, and other groups have developed or are developing frameworks and software for managing grid computing projects. The European Community (EU) is sponsoring a project for a grid for high-energy physics, earth observation, and biology applications. In the United States, the National Technology Grid is prototyping a computational grid for infrastructure and an access grid for people.

Grid computing appears to be a promising trend for three reasons: (1) its ability to make more cost-effective use of a given amount of computer resources, (2) as a way to solve problems that can't be approached without an enormous amount of computing power, and (3) because it suggests that the resources of many computers can be cooperatively and perhaps synergistically harnessed and managed as a collaboration toward a common objective. In some grid computing systems, the computers may collaborate rather than being directed by one managing computer. One likely area for the use of grid computing will be pervasive computing applications - those in which computers pervade our environment without our necessary awareness.

The establishment, management, and exploitation of dynamic, cross-organizational sharing relationships require new technology. This technology is Grid architecture and supporting software protocols and middleware [1, 2, 7, 9, 13, 14, 15, 16, 17, 18, 20]

2.3.1 Globus Toolkit

The Globus Project [2] provides software tools that make it easier to build computational grids and grid-based applications. These tools are collectively called The Globus Toolkit. The Globus Toolkit is used by many organizations to build computational grids that can support their applications.

The composition of the Globus Toolkit can be pictured as three pillars: Resource Management, Information Services, and Data Management. Each pillar represents a primary component of the Globus Toolkit and makes use of a common foundation of security. GRAM implements a resource management protocol, MDS implements an information services protocol, and GridFTP implements a data transfer protocol. They all use the GSI security protocol at the connection layer [2, 20].

GRAM [1, 2, 26] is designed to provide a single common protocol and API for requesting and using remote system resources, by providing a uniform and flexible interface to local job scheduling systems. The Grid Security Infrastructure (GSI) provides mutual authentication of both users and remote resources using GSI (Grid-wide) PKI-based identities. GRAM provides a simple authorization mechanism based on GSI identities and a mechanism to map GSI identities to local user accounts.

MDS [1, 2, 27, 28] is designed to provide a standard mechanism for publishing and discovering resource status and configuration information. It provides a uniform and flexible interface to data collected by lower-level information providers. It has a decentralized structure that allows it to scale, and it can handle static (e.g., OS, CPU types, system architectures) or dynamic data (e.g., disk availability, memory availability, and loading). A project can also restrict access to data by combining GSI (Grid Security Infrastructure) credentials and authorization features provided by MDS.

GridFTP [1, 2, 23, 24, 25] is a high-performance, secure, reliable data transfer protocol optimized for high-bandwidth wide-area networks. The GridFTP protocol is based on FTP, the highly-popular Internet file transfer protocol. GridFTP provides the following protocol features: 1, GSI security on control and data channels. 2, Multiple data channels for parallel transfers. 3, Partial file transfers. 4, Direct server-to-server transfers. 5, Authenticated data channels. 6, Reusable data channels. 7, Command pipelining

2.3.2 MPICH-G2

MPI is a message-passing library standard that was published in May 1994. The "standard" of MPI is based on the consensus of the participants in the MPI Forums [3], organized by over 40 organizations. Participants include vendors, researchers, academics, software library developers and users. MPI offers portability, standardization, performance and functionality [22].

The advantage for the user is that MPI is standardized on many levels. For example, since the syntax is standardized, you can rely on your MPI code to execute under any MPI implementation running on your architecture. Since the functional behavior of MPI calls is also standardized, your MPI calls should behave the same regardless of the implementation. This guarantees the portability of your parallel programs. Performance, however, may vary between different implementations.

MPICH-G2 [4, 5] is a grid-enabled implementation of the MPI v1.1 standard. That is, using services from the Globus Toolkit® (e.g., job startup, security), MPICH-G2 allows you to couple multiple machines, potentially of different architectures, to run MPI applications. MPICH-G2 automatically converts data in messages sent between machines of different architectures and supports multiprotocol communication by automatically selecting TCP for intermachine messaging and (where available) vendor-supplied MPI for intramachine messaging. Existing parallel programs written for MPI can be executed over the Globus infrastructure just after recompilation [19].

2.3.3 SUN Grid Engine

Sun Grid Engine is new generation distributed resource management software which dynamically matches users' hardware and software requirements to the available (heterogeneous) resources in the network, according to policies usually defined by management.

Sun Grid Engine acts as the central nervous system of a cluster of networked computers. Via so-called daemons, the Grid Engine Master supervises all resources in the network to allow full control and achieve optimum utilization of the resources available.

Sun Grid Engine aggregates the compute power available in dedicated compute farms, networked servers and desktop workstations, and presents a single access point to users needing compute cycles. This is accomplished by distributing computational workload to available systems, simultaneously increasing the productivity of machines and application licenses while maximizing the number of jobs that can be completed.

In addition, Sun Grid Engine software helps lower the costs of purchasing, installing, setting up and administering the computing environment because it allows: Maximized use of new/existing resources, Lower administration costs, Lower upgrade costs, More efficient reuse of existing legacy, Resources [6, 22].

3 Hardware and Software Configuration

The test environment, described in the next table, we build 2 clusters to form a multiple cluster environment. Each cluster has two slave nodes and one master node. Each nodes are interconnected through 3COM 3C9051 10/100 Fast Ethernet Card to Accton CheetahSwitch

AC-EX3016B Switch HUB; Each master node is running SGE QMaster daemon and SGE execute daemon to running, manage and monitor incoming job and Globus Toolkit v2.4. Each slave node is running SGE execute daemon to execute income job only.

Cluster 1			
Hostname	Grid	Grid1*	Grid2
FQDN	grid.hpc.csie.thu.edu.tw	grid1.hpc.csie.thu.edu.tw	Grid2.hpc.csie.thu.edu.tw
IP	140.128.101.172	140.128.101.188	140.128.101.188
CPU	Intel Pentium 3 - 1Ghz *2	Intel Celeron 1.7GHz	Intel Celeron 1.7GHz
RAM	512MB SDRAM	768MB DDR RAM	256MB DDR RAM

Cluster 2			
Hostname	Grid3*	Grid4	Grid5
FQDN	grid3.hpc.csie.thu.edu.tw	grid4.hpc.csie.thu.edu.tw	grid5.hpc.csie.thu.edu.tw
IP	140.128.101.187	140.128.101.188	140.128.101.189
CPU	Intel Celeron 1.7GHz	Intel Pentium 3 - 866Mhz *2	Intel Pentium 4 - 2.53GHz
RAM	256MB DDR RAM	1.5GB DDR RAM	256MB DDR RAM

* stand for Master node of the cluster, the others is slave node.

Table 1. Hardware Configuration

4 Experimented Result

The experiment consists of three scenarios: single Personal Computer, Cluster environment and Grid environment. First step, we run a MPI program on a PC or PC-based SMP system to evaluate the system performance. Second step, we connect 3 Personal Computer together to form a Cluster environment (In our testbed is Cluster1 and Clutser2) Then, running the same MPI program to evaluate the system performance. Third step, through the WAN connection, we connect the Cluster1 and Cluster2 together to form a grid environment. Then, the same MPI program is executed to evaluate the system performance.

4.1 Compute PI

It computes the value of π by numerical integration. Since

$$\int_0^1 \frac{1}{1+x^2}dx = \tan^{-1}(1) = \frac{\pi}{4}$$

We can compute π by integration the function $f(x) = 4/1+x^2$ from 0 to 1. We compute an approximation by dividing the interval [0, 1] into some number of subintervals and then computing the total area of these rectangles by having each process compute the areas of some subset.

The subintervals are 1,000,000,000 (1 billion) in our experiment.

	Grid	Grid1	Grid2	Grid3	Grid4	Grid5
Single PC	43.239s	98.542s	95.239s	98.344s	49.873s	50.091s
Cluster Environment	25.756s			26.461s		
Grid Environment	12.968s					

4.2 Prime Number

For example, if you want to find the prime numbers between 1 and 20,000,000 (20 million). It proceeds to write code that initially runs on a lead node and sends the task of testing 101-200 to node 1, and sends the task of testing 201-300 to node 2, and so on. Along with the testing task, there would also be an instruction to return whatever primes a slave node discovered to the lead node. When all nodes have completed their tasks, there will have a message to tell you how many prime be found and what the biggest prime number is.

In our experiment, we try to fine the prime numbers between 1 and 20,000,000 (20 million).

	Grid	Grid1	Grid2	Grid3	Grid4	Grid5
Single PC	43.139s	53.089s	51.512s	53.145s	49.817s	34.416s
Cluster Environment	21.541s			24.833s		
Grid Environment	12.411s					

4.3 Matrix Multiplication

The matrix operation derives a resultant matrix by multiplying two input matrices, x and y, where matrix x is a matrix of N rows by P columns and matrix y is of P rows by M columns. The resultant matrix c is of N rows by M columns. The serial realization of this operation is quite straightforward as listed in the following:

```
for (k=0;k<M;k++)
```

```
for (i=0;i<N;i++) {
    c[i][k]=0.0;
    for (j=0;j<P;j++)
    c[i][k]+=a[i][j]*b[j][k]; }
```

Its algorithm requires n^3 multiplications and n^3 additions, leading to a sequential time complexity of $O(n^3)$. Use parallel technique to get the fastest multiplication speed.

The problem sizes were 1024×1024 in our experiment.

	Grid	Grid1	Grid2	Grid3	Grid4	Grid5
Single PC	80.102s	117.978s	115.526s	118.125s	67.292s	110.367s
Cluster Environment	41.008s			35.133s		
Grid Environment	23.241s					

5 Conclusions and Future Work

In this paper, we construct a grid computing environment testbed on multiple Linux PC Clusters by using Globus Toolkit (GT) and SUN Grid Engine (SGE). Then, we execute some MPI programs on ours testbed: PI, Prime problem, Matrix multiplication to evaluate the system performance. The experiment consists of three scenarios: Single PC, Cluster computing environment and Grid computing environment. After the experiment, we compare the different system environment performance. As the experiment result, single PC although can process the same problem like Cluster or Grid can do, but the processing time is always slow than the cluster or grid. Cluster Computing, it consists several of PC to form a Cluster system. So, it is obvious that Cluster's computing performance is better than single PC. Cluster, when a job is submitted to a master node of the cluster, then the master node will divide the incoming job into several part and send the subjob to the other slave nodes synchronously and cooperatively solve the problem. As the experiment result, we can find from the Single PC to Cluster system, we gain 50% performance improvement. Finally, we use our idea to connect multiple Linux PC Clusters to form a Grid Computing environment and run the same problem on. As the experiment, the performance improves 50% than cluster computing experiment. After this paper, it is no doubt that Grid computing technology will become a new way to tackle complex problem from the scientific, engineering to biotechnology. High-Speed computing performance makes every complicated thing easy. In our future work, we will extend the Grid testbed for job execution by job scheduling policy to efficiently use CPU idle time. Make the system fully utilized.