Grid Computing – the Hasty Computing to Access Internet

A.Ashok[1], Harikrishnan.N[2], Thangavelu.V[3]

Bit Campus, Anna University- Trichy
[1]ashokannadurai@gmail.com, [2]hariever4it@gmail.com, [3]thangavelc@gmail.com

Abstract

Today we are in the Internet world and everyone prefers to enjoy fast access to the Internet. But due to multiple downloading, there is a chance that the system hangs up or slows down the performance that leads to the restarting of the entire process from the beginning. This is one of the serious problems that need the attention of the researchers.

So we have taken this problem for our research and in this paper we are providing a layout for implementing our proposed Grid Model that can access the Internet very fast. By using our Grid we can easily download any number of files very fast depending on the number of systems employed in the Grid. We have used the concept of Grid Computing for this purpose.

The Grid formulated by us uses the standard Globus Architecture, which is the only Grid Architecture currently used world wide for developing the Grid. And we have proposed an algorithm for laying our Grid Model that we consider as a blueprint for further implementation. When practically implemented, our Grid provides the user to experience the streak of lightening over the Internet while downloading multiple files.

Keywords: *Grid Security Interface (GSI), Global Access to Secondary Storage (GASS), Monitoring and Discovery Service (MDS), Globus Resource Allocation Manager (GRAM).*

INTRODUCTION

What's Grid computing? Grid Computing is a technique in which the idle systems in the Network and their “ wasted “ CPU cycles can be efficiently used by uniting pools of servers, storage systems and networks into a single large virtual system for resource sharing dynamically at runtime. These systems can be distributed across the globe; they're heterogeneous (some PCs, some servers, may be mainframes and supercomputers); somewhat autonomous (a Grid can potentially access resources in different organizations).

Although Grid computing is firmly ensconced in the realm of academic and research activities, more and more companies are starting to turn to it for solving hard-nosed, real-world problems.

IMPORTANCE OF GRID COMPUTING

Grid computing is emerging as a viable technology that businesses can use to wring more profits and productivity out of IT resources -- and it's going to be up to you developers and administrators to understand Grid computing and put it to work.

It's really more about bringing a problem to the computer (or Grid) and getting a solution to that problem. Grid computing is flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions, and resources. Grid computing enables the virtualization of distributed computing resources such as processing, network bandwidth, and storage capacity to create a single system image, granting users and applications seamless access to vast IT capabilities. Just as an Internet user views a unified instance of content via the World Wide Web, a Grid user essentially sees a single, large, virtual computer.

Grid computing will give worldwide access to a network of distributed resources - CPU cycles, storage capacity, devices for input and output, services, whole applications, and more abstract elements like licenses and certificates.

For example, to solve a compute-intensive problem, the problem is split into multiple tasks that are distributed over local and remote systems, and the individual results are consolidated at the end. Viewed from another perspective, these systems are connected to one big computing Grid. The individual nodes can have different architectures, operating systems, and software versions. Some of the target systems can be clusters of nodes themselves or high performance servers.

TYPES OF GRID

The three primary types of grids and are summarized below:

- **Computational Grid**

  A computational grid is focused on setting aside resources specifically for computing power. In this type of grid, most of the machines are high-performance servers.
- **Scavenging grid**

  A scavenging grid is most commonly used with large numbers of desktop machines. Machines are scavenged for available CPU cycles and other resources. Owners of the desktop machines are usually given control over when their resources are available to participate in the grid.
- **Data Grid**

  A data grid is responsible for housing and providing access to data across multiple organizations. Users are not concerned with where this data is located as long as they have access to the data.

OUR PROPOSED GRID MODEL

We are using the Scavenging Grid for our implementation as large numbers of desktop machines are used in our Grid and later planning to extend it by using both Scavenging and data Grid. Figure1 gives an idea about the Grid that we have proposed.

PROBLEMS DUE TO MULTIPLE DOWNLOADING

While accessing Internet most of us might have faced the burden of multiple downloading and in particular with downloading huge files i.e., there can be a total abrupt system failure while a heavy task is assigned to the system. The system may hang up and may be rebooted while some percentage of downloading might have been completed. This rebooting of the system leads to download of the file once again from the beginning, which is one of the major problems everyone is facing today.

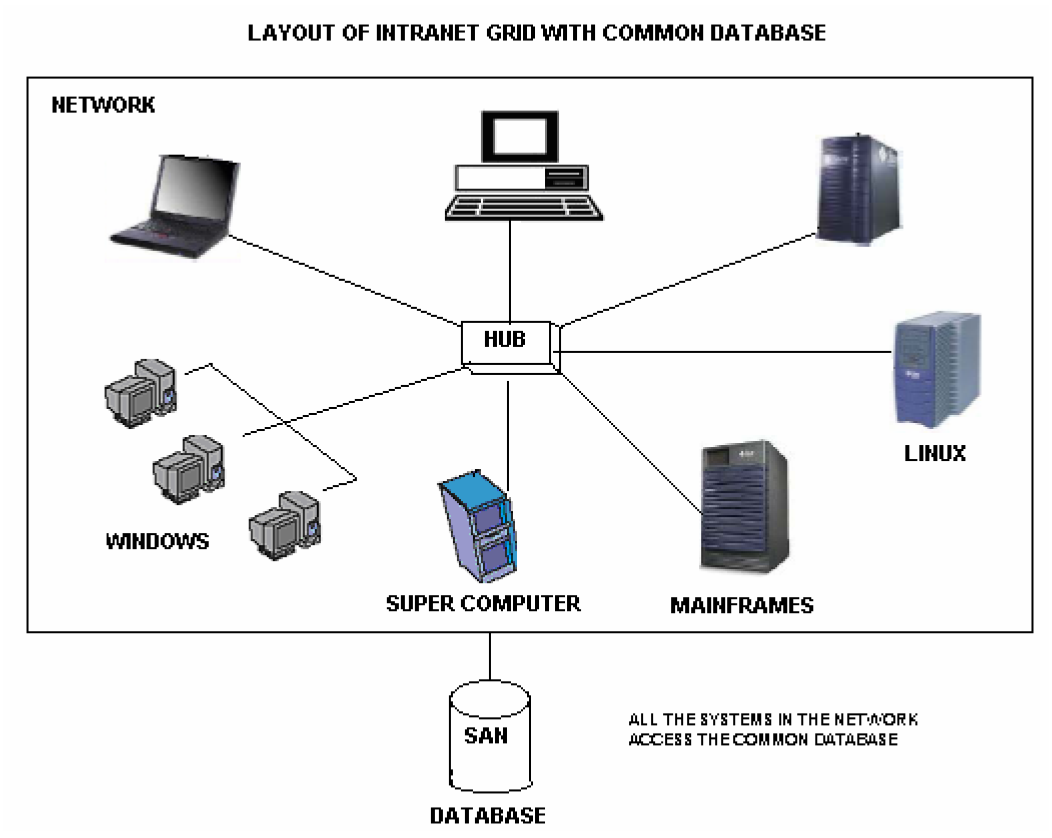
Let us consider N numbers of files of different sizes (in order of several MBs) are being downloaded on a single system (a PC). This will take approximately some minutes or even some hours to download it by using an Internet connection of normal speed with a single CPU. This is one of the tedious tasks for the user to download multiple files at the same time. Our Grid plays a major role here.

CONCEPT OF OUR PROPOSED GRID

In order to avoid this problem we have formulated our own Grid for such an access to the Internet via an Intranet (LAN). By using our Grid these large numbers of files are distributed evenly to all the systems in the Network by using our Grid.

For example we have taken into account of a small LAN that consists of around 20 systems out of which 10 systems are idle and 5 systems are using less amount of CPU(for our consideration) and their CPU cycles are wasted. And our work begins here, as we are going to efficiently utilize those “wasted CPU cycles” into “working cycles”.

FIGURE 1: LAYOUT OF OUR INTRANET GRID


LAYOUT OF INTRANET GRID WITH COMMON DATABASE
NETWORK
HUB
LINUX
WINDOWS
SUPER COMPUTER
MAINFRAMES
SAN
DATABASE
ALL THE SYSTEMS IN THE NETWORK ACCESS THE COMMON DATABASE

WORKING OF THE PROPOSEDGRID

When we are downloading multiple files using Internet the Grid formulated by us comes in to action. A dialog box will appear on the Desktop asking the user whether to use the Grid or not? If the user selects “use the Grid”, then automatically the available system resources in the Network are obtained by the Globus Toolkit. The configurations of the idle systems are noted and the highest configuration system gets the highest priority in the priority Queue.

E.g. If there is a supercomputer with 8 CPUs, another Supercomputer with 5 CPUs and some other PCs with P3-2.0GHz, P4-2.0GHz, P4-2.5GHz, P3-1.0GHz, P3-1.3GHz, P4-1.5GHz, P3-1.13GHz, P4 -2.4GHz are found in the Network.

Then the order of priority will be: 1. Supercomputer-8 CPUs, 2. Supercomputer-5 CPUs, 3. P4-2.5GHz, 4. P4-2.4GHz, 5. P4-2.0GHz, 6. P3-2.0GHz, 7. P4-1.5GHz, 8. P3-1.3GHz, 9. P3-1.13GHz, 10. P3-1.1GHz.

Now the user can click any number of files to download. The file size of each file is obtained and is stored in the priority Queue based on maximum size as highest priority. Now the highest priority file is matched with the highest priority system in the Network. The files get evenly distributed to their matched “idle systems”. The downloading gets completed in those systems and these file gets stored in the common database. The authenticated user can access this database and can retrieve his file that he has downloaded.

The various processes that are taking place in our Grid such as authentication, availability of resources, scheduling, data management and finally job and resource management are viewed by following a standard architecture – The Globus Architecture.

EMPLOYING THE GLOBUS ARCHITECTURE IN OUR GRID

While planning to implement a Grid project, we must address issues like security, managing and brokering the workload, and managing data and resources information. Most Grid applications contain a tight integration of all these components.

The Globus Project provides open source software tools that make it easier to build computational Grids and Grid-based applications. These tools are collectively called the Globus Toolkit. Globus Toolkit is the open source Grid technology for computing and data Grids. On the server side, Globus Toolkit 2.2 provides interfaces in C. On the client side, it provides interfaces in C, Java language, and other languages. On the client side, the Java interfaces are provided by the Java Commodity Grid (CoG) Kit. Globus runs on Linux, AIX, HP-UX, Solaris, and also on windows operating systems.

The Globus architecture represents a multiple-layer model. The local services layer contains the operating system services and network services like TCP/IP. In addition, there are the services provided by cluster scheduling software (like IBM Load Leveler) -- job-submission, query of queues, and so forth. The cluster scheduling software allows a better use of the existing cluster resources. The higher layers of the Globus model enable the integration of multiple or heterogeneous clusters.

ACCESSING THE INTRANET GRID:

When any user wants to access our proposed Intranet Grid in order to download multiple files over the Internet, then he should follow certain procedures that we consider necessary for the security of our Grid. The main Requirements for

Processing in Grid Environment are:

- Security: single sign-on, authentication, authorization, and secure data transfer.
- Resource Management: remote job submission and management.
- Data Management: secure and robust data movement.
- Information Services: directory services of available resources and their status.
- Fault Detection: Checking the intranet.
- Portability: C bindings (header files) needed to build and compile programs.

<u>Existing Algorithm for Globus Architecture</u>:

```
Step [1].Create security_proxy via GSI services
Step [2].Access a MDS-GIIS server
Step [3].Search for required machine(s)
Step [4].Rank the machine list based on a scheduling policy
Step [5].Prepare the data
Step [6].Transfer the data to the target machine by using
         GASS services
```

```
Step [7].Prepare a RSL document
Step [8].Submit the program using GRAM services
Step [9].React to status changes from GRAM
Step [10]. Get results via GASS
```

Here, we have got the resources available in the Network which is automatically done by have the Globus Toolkit in the server. When we want to download a file this information has to be matched with the client module and then the downloading has to be carried out in the clients. For this we have added some modules to the Grid Architecture.

Added module:

```
Step [11]. Get the Information about files to be downloaded.
Step [12]. Match the files with appropriate Machines.
Step [13]. Store files in common database.
Step [14]. Retrieval of data from database is done after proper
           authentication.
```

You'll also see how Grid services and the very framework it all rests on is very much like object-oriented programming.

PROPOSED ALGORITHM FOR OUR INTRANET GRID

Steps to perform multiple downloading on the Grid. The host details are got from the server of the LAN in order to identify the various hosts. The host information is got whenever needed on the priority queue basis.

```
//module for downloading files
    [1] Start lookup // look for file size and resource information
    [2] Declare nres, nfile // no of resources available and no of files
    [3] Input nres, nfiles
    [4] Input size // the file size
    [5] Initialize P1 → res info // store the resource information in priority queue
        P1 with highest system configuration as priority
    [6] Initialize P2 → file size // store the file information in the priority queue P2
        with  maximum file size as priority
    [7] If condition (nfiles == nres) // check whether the no of resources is equal
        to no. of files
    [8] Initialize counter
    [9] For (counter =1; counter <= nres; counter++) // initialize the loop to assign
        the files.
    [10]  Assign the 1st file of P2 to the 1st node in P1.// first node will be node with
          highest configuration and first file will be the file  maximum size.
    [11]  Start processing // files directed to the appropriate system for accessing
          their wasted CPU cycles.
    [12] Loop
    [13] Else:
    [14] Start timer
    [15] Delay → 1 min
    [16]  Collect incoming files // the files that the user clicked to download in this
          duration.
    [17] Assign the files → P2
    [18] Goto step 8
    [19] Goto step 1
    [20] End // when the user exits from proposed Grid.
```

CONCLUSION

Grid computing was once said to be fading out but due to the technological convergence it is blooming once again and the Intranet Grid we have proposed adds a milestone for the Globalization of Grid Architecture, which, leads to the hasty computing that is going to conquer the world in the nearest future. By implementing our proposed Intranet Grid it is very easy to download multiple files very fast and no need to worry about the security as we are authenticating each and every step taking place in our Grid and in particular user to access the database. Further implementations could be carried out in the nearest future.

Article received: **2009-01-23**