International Journal of Computer and Communication Technology

Volume 5 | Issue 1 Article 4

January 2014

Performance Evaluation of Computation Intensive Tasks in Grid

P. Raghu
Department of Computer Technology and Applications Coimbatore Institute of Technology Coimbatore, India, raghu_17688@yahoo.com

K. Sriram
Department of Computer Technology and Applications Coimbatore Institute of Technology Coimbatore, India, ksriram89@gmail.com

Recommended Citation
Raghu, P. and Sriram, K. (2014) "Performance Evaluation of Computation Intensive Tasks in Grid," *International Journal of Computer and Communication Technology*: Vol. 5 : Iss. 1 , Article 4.
DOI: 10.47893/IJCCT.2014.1214
Available at: https://www.interscience.in/ijcct/vol5/iss1/4

Performance Evaluation of Computation Intensive Tasks in Grid

P.Raghu, K. Sriram
Department of Computer Technology and Applications
Coimbatore Institute of Technology Coimbatore, India
raghu_17688@yahoo.com, ksriram89@gmail.com

Abstract- Grid computing is a special type of parallel computing, which allows us to unite pools of servers, storage systems, and networks into a single large virtual super computer. Grid computing has the advantages of solving complex problems in a shorter time and also makes better use of the existing hardware. It can take advantage of underutilized resources to meet business requirements while minimizing additional costs.

There are many Grid setup tools available. In this paper, Globus Toolkit, an open source tool for grid enabled applications, is considered. Initially grid is established between two systems running Linux, using Globus Toolkit. A simple matrix multiplication program, which is capable of running both in grid and stand alone systems, is developed. The application is executed in single system varying the order of the matrices. The same application is split into two sub jobs and run on two grid machines with different orders. Finally the results of the execution are compares and the results are presented in graphs. The work can be extended further to find the type of parallelizing suitable for the application developed. Similarly, FP tree algorithm is taken and the data sets are fed into different machine and in stand alone system. A suitable load balancing mechanism for grid application is discussed.

The sections in the paper are arranged as following; Introduction to Grid, Grid setup using Globus toolkit, splitting of the matrix application, FP tree algorithm, performance results, future works, conclusion and references.

I. INTRODUCTION

Computing is becoming distributed and individual users or client applications gain access to computing resources such as processors or storage or data as needed. The user has little or no knowledge of where the resources are located or what the underlying technologies, hardware, OS and so on are.

Grid is an infrastructure, which allows integrated, collaborative use of geographically separated, autonomous resources. Grid computing allows us to use the efficiency of multiple systems resources to a single user point for any large scale application. Grid computing is, simply, linking together of multiple computing resources, including processors and storage devices, to provide a much larger, more powerful single virtual computer. To a user, or an application, the system is viewed as a single enormous virtual computing system. Grid computing allows us to unite pools of servers, storage systems, and networks into a single large system so we can deliver the power of multiple-systems resources to a single user point for a specific purpose.

Grid computing is the next logical step in distributed networking. Just as the Internet allows users to share ideas and files as the seeds of projects, grid computing lets us share the resources of disparate computer systems. Grid computing takes the ability for computers to communicate a step further. With grid computing, we can reach out and use computational or storage resources on machines other than our own. Regardless of the depth of a grid's resources, all the grid user experiences is the processing resources of a very large virtual computer.

In general, grid computing uses parallel and distributed processing to dynamically share and aggregate multiple computer resources. Typically, arrays of computational power are constructed from a network of computers. The network of computers may consist of several computers in a single department, or it may consist of hundreds or even thousands of computers spanning multiple geographic locations. Applications are submitted to the grid, and a piece of software called "middleware" decides which specific machine in the grid should execute the application. This determination is made at run time and is based on the software and hardware requirements of the application as well as the availability of the computers in the grid.

Grid computing offers a variety of benefits, including: making individual applications run much faster, allowing multiple applications to share computing resources, increasing the use of all available computing resources, providing flexibility to add more resources to the grid as needed, increasing productivity by providing users the resources they need on demand, using existing resources more efficiently, responding quickly to changing business and market demands, enabling collaboration among dispersed entities and creating virtual organizations that can share resources and data utilization of existing resources. [1]

Grid computing can be seen as a journey along a path of integrating various technologies and solutions that move us closer to the final goal. Its key values are in the underlying distributed computing infrastructure technologies that are evolving in support of cross organizational application and resource sharing--in a word, virtualization—virtualization across technologies, platforms, and organizations. This kind of virtualization is only achievable through the use of open standards. [3]

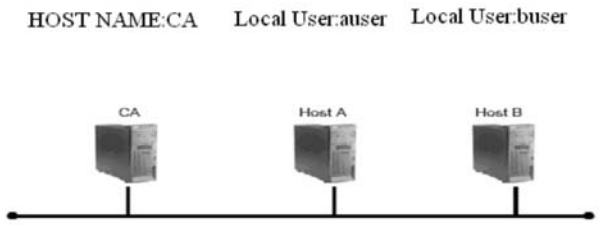

Fig.1 Setup of grid environment

There are many tools available for setting up the grid environment. In this paper we used Globus, atoolkit addressing key technical problems in the development of Grid enabled tools, services, and applications. The Globus Toolkit of the Globus- Alliance is a Middleware for Grid Systems. The Globus Toolkit 4 offers all necessary components for the implementation of Grid systems.

We used three hosts out of which two were Grid nodes for running grid applications and a CA (Certificate Authority) host for providing certificates to grid nodes. Hence the systems used are hosta, hostb and CA. As Globus toolkit is targeted for linux based systems all the three hosts were running Redhat Enterprise Linux 5.0. The local users required are globus(in all three nodes for globus installation and administration), a-user(hosta) and b-user(hostb).The a-user and b-user are grid users who can submit the job in grids.

We installed java SDK and apache-initially as Globus toolkit had those dependencies. As multiple systems are involved in Grid, we should make sure that time is synchronized in all the machines. GSI certificates use time stamps and are very sensitive to the time. If the system time of the grid environment is not set correctly, errors might occur on the use of GSI certificates [1]. Hence we used NTP protocol to synchronise the time in Grid environment. Next we installed Globus toolkit source package and configured the environment variables.[5] All the systems in the Grid, should have a valid host certificate and all the local users who submit jobs in grid should have a user certificate signed by the CA host.

Hence we requested host and user certificates, signed them using CA host and placed them in suitable directories of the grid nodes for proper functioning of the grid environment.

Finally we configured java ws core, GridFTP and RFT which are parts of Globus toolkit. GridFTP is a protocol for sending and receiving files within the grid environment in a secure and reliable way [3].

II. SPLITTING OF MATRIX APPLICATION

Consider the following matrix multiplication. It is the traditional way of computing a matrix multiplication.

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Fig.2 Matrix Multiplication

Consider fig.3, in which the first matrix and first column of B matrix is alone enough for getting the first column output.

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 & \bullet \\ 3 & \bullet \end{pmatrix} = \begin{pmatrix} 1 & \bullet \\ 3 & \bullet \end{pmatrix}$$

Fig.3 Masking second column

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} * \begin{pmatrix} \bullet & 2 \\ \bullet & 4 \end{pmatrix} = \begin{pmatrix} \bullet & 2 \\ \bullet & 4 \end{pmatrix}$$

Fig.4 Masking first column

Likewise in fig.4, the first matrix and second column of B matrix is alone enough for getting the second column output. Hence we splitted the n columns of the given matrix into two and submitted for execution in two grid systems simultaneously. Hence the entire first matrix and first (n/2) columns of second matrix is given to hosta and the entire first matrix and second (n/2) columns of B matrix is given to hostb. Since both the execution takes place simultaneously, time complexity is reduced. If the job is splitted between more than two systems time consumed will be still less. A perfectly scalable application will finish in one tenth of the time, if it uses ten times the number of processors.[1]

The section(IV. PERFORMANCE SECTION) gives the results of execution of the application in single system and also in the grid nodes. The performance results show that when the size of the process or application increases grid approach seems to be more efficient.

III. FPTREE ALGORITHM

Most of the association rule mining algorithms suffer from the following two shortcomings:[4]

1. It is costly to handle large number of candidate sets. For instance, if there are 10^4 frequent 1-itemsets, then approximately, 10^7 candidate 2-itemsets are generated. Moreover, if there is a frequent set of size 100, then roughly 10^{30} candidate sets are generated in this process.
2. It is tedious to repeatedly scan the database and check a large set of candidates by pattern matching.

Keeping this in mind, a new class of algorithms has recently been proposed which avoids the generation of large number of candidate sets. We describe one such method, called the FP-Tree Growth algorithm.

A frequent pattern tree (or FP-tree) is a tree structure consisting of an item-prefix-tree and a frequent-item-header table.

- Item-prefix-tree:
- It consists of a root node labeled null
- Each non-root node consists of three fields:
  - Item ame,
  - Support Count
  - Node link
- Frequent-item-header -table: It consists of two fields;
  - Item name
  - Head of node link which points to the first node in the FP-Tree carrying the item name.

It may be noted that the FP-tree is dependent on the support threshold ó. For different values of ó, the trees are different.

IV. OBSERVATION AND RESULTS

[MATRIX APPLICATION]

	ORDER	EXECUTION TIME(ms)
1	50	6
2	100	7
3	200	121
4	300	477
5	500	2457
6	1000	23313
7	1500	90070
8	2000	325124

Table I: Traditional Approach

Table I, gives the execution timings of the entire application in a single system which is a traditional approach. The application is executed in a single system which had 256 MB RAM and 30 GB hard disk with pentium 4 processor. The order of the matrices varied from 50 to 2000 and the time taken for each order is tabulated in milliseconds.

Table II, gives the execution timings of the application in a grid systems. It is found that whenever the job is intensive, Grid computing is efficient. The application is executed in the two available grid systems and each system had a 256 MB RAM and 30 GB hard disk with pentium 4 processor. The order of the matrices were varied from 50 to 2000 and the time taken for each order is tabulated in milliseconds.

	ORDER	EXECUTION TIME(ms)
1	50	6
2	100	11.5
3	200	62
4	300	201.5
5	500	1197
6	1000	10883.5
7	1500	41713.5
8	2000	133781

Table II: Grid Approach

Fig.5, shows the graph drawn with the data from Table I and fig.6, shows the data drawn from the Table II. The time taken for large tasks is nearly reduced to half and even more.The scale consideration in Y-axis is upto 350000 ms whereas in fig.6, it is only upto 16000 ms which is nearly half of the traditional approach.

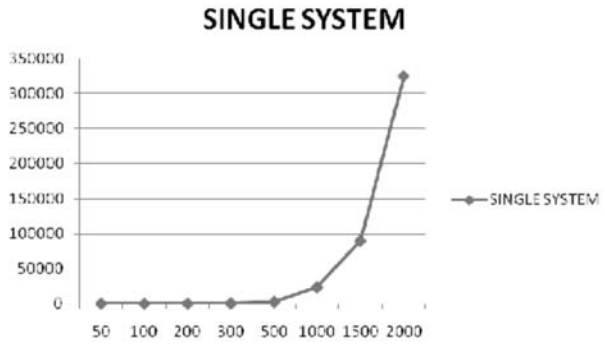

Fig.5 Single system execution results

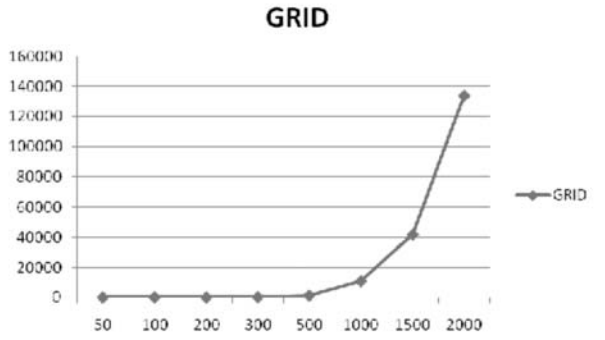

Fig.6 Grid approach execution results

Hence we can arrive at a conclusion that the time taken for completion of any complex job, in the grid environment is dependant on the number of processors among which the job is divided. Here as only two processors are used, we achieved half of the time efficiency when compared to the single system approach. Hence the number of systems needed for the application is purely dependant on the application itself. Fig.7, gives the bar chart showing the comparision of the results obtained.

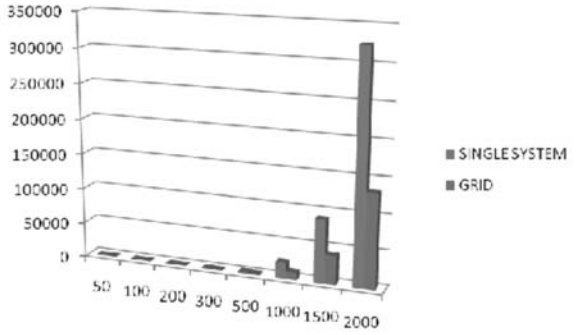

Fig.7 Comparision using bar chart

[FP TREE ALGORITHM]

Let the number of data sets be *d* and number of hosts be *h*. In the Grid environment [d/3] data sets is split to [h/3] hosts. If there are 3 data sets and 3 hosts, each host will compute one data set.

As this project emphasizes on comparing the time complexity of the task in a single system and grid scenario, the execution time of the FP tree algorithm was recorded with respect to various datasets.

S.No	No. of Data Sets	Execution Time (ms) Single System Approach	Grid Approach
1	3 Data Sets	31797	20784
2	6 Data Sets	92540	61118
3	9 Data Sets	118894	70277

Table III: Execution Time in Single System and Grid Environment

The above results are compared using graphs.

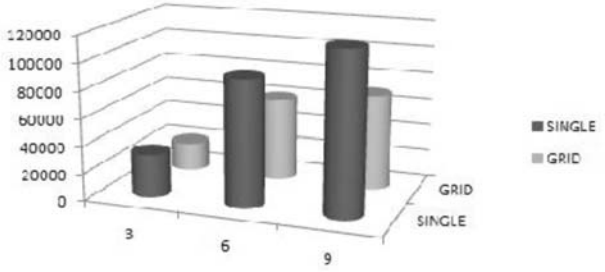

Fig.8 Comparison using bar chart

IV. LOAD BALANCING

We know that there are two load balancing mechanisms available in grid computing: static and dynamic. In the static case the mechanisms related to load balance are made at compile time when resource requirements are estimated. In dynamic load balancing the resources are allocated or reallocated at run time, which may determine whose and when tasks are migrated. Many load balancing algorithms suffer from the following shortcomings: Heterogeneity, Scalability, and Adaptability.

Grid Computing is a finite set of G clusters Ck inter connected by gates Gk.Each cluster contains sites Interconnected by switches. A tree based model could be built as follows:

Create two level sub trees for each site. The leaf of the sub tree corresponds to computing element of the sub tree and root of the sub tree is the virtual node associated to the site. Virtual node will manage the work load of the site. Sub trees of all sites of the clusters are aggregated to form three level sub tree. These sub trees are connected to form a four level sub tree. The virtual node manages it sites. Nodes associated with physical clusters manage the work load of computing elements. At leaves we find that the computing elements of grid linked to respective sites and clusters. General strategy:

(i) Estimate the current work load of a cluster/site periodically from its elements.
(ii) Compute the Standard deviation over the work loaded index in order to measure the deviation between its involved elements.
(iii) Send work load information to work load manager.
(iv) Measure the imbalance /saturation state.
(v) We do portioning based on overloaded and under loaded ones. When supply is more than demand we do task transfer regarding communication cost included by this transfer.

V. FUTURE WORKS

The results show that Grid approach is more suitable for computation intensive jobs. Many other intensive jobs can be submitted in grids and their performance can be measured. Based on their performance results, certain parameters can be derived. Such derived parameters can be used to decide whether to parallelize data or algorithm for the given application. Such derived parameters can also be used to decide the optimal number of hosts for the given application.The work is to be extended on choosing an optimal scheduling and load balancing techniques.

VI. CONCLUSION

Thus, the results show that grid computing can be used for any complex task solving, in any fields. Grid computing enables organizations to take advantage of various computing resources in ways not previously possible. They can take advantage of underutilized resources to meet business requirements while minimizing additional costs. The nature of a computing grid allows organizations to take advantage of parallel processing, making many applications financially feasible as well as allowing them to complete sooner.

VII. REFERENCES

[1] Bart Jacob, Michael Brown, Kentaro Fukui and Nihar Trived, “Introduction to grid computing”.
[2] Luis Ferreira, Viktors Berstis, Jonathan Armstrong, Mike Kendzierski, Andreas Neukoetter, Masanobu Takagi, Richard Bing, Wo, Adeeb, Amir,Ryo Murakawa, Olegario Hernandez, James Magowan, Norbert Bieberstein, “Introduction to Grid Computing with Globus”.
[3] Bart Jacob, Luis Ferreira, Norbert Bieberstein, Candice (Steve) Yu, “Enabling Applications for Grid Computing with Globus”.
[4] Data Mining Techniques,Second Edition, Arun K Pujari,Universities Press,Second Edition
[5] www.globus.org/alliance/papers/UKglobus-v4.pdf