ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

Process Scheduling Approach for Starvation Improvement with Time Delay Analysis in Grid Resources Allocation

Er. Himanshu Jain [1], Kavita Khatkar[2]

M.Tech, CSE, JCDM College of Engineering, Sirsa, India [1]

Asst Professor, CSE, JCDM College of Engineering, Sirsa, India [2]

Abstract: Grid computing has been a buzzword in information technology since past few years. Grid computing is an infrastructure involving collaboration of computers, databases & network resources available, to perform manipulation of intensive and large scale data set problems. The hike in the complexities of computational problems in modern era of science and technology forced the engineers and scientists to cross the organizational boundaries to get desired data manipulation. The best logical solution to this issue is creating Cache to reduce load on each grid to improve average waiting and response time.

Keywords: VO (virtual organization), ELBDGBJS (Efficient Load Balancing Dynamic Grouping based Job Scheduling).

I. INTRODUCTION

The real and specific problem that underlies the Grid concept is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations. The sharing that we are concerned with is not primarily file exchange but rather direct access to computers, software, data, and other resources, as is required by a range of collaborative problem-solving and resource-brokering strategies emerging in industry, science, and engineering. This sharing is, necessarily, highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. A set of individuals and/or institutions defined by such sharing rules form what we call a virtual organization (VO)

VOs vary tremendously in their purpose, scope, size, duration, structure, community, and sociology. Nevertheless, careful study of underlying technology requirements leads us to identifya broad set of common concerns and requirements. In particular, we see a need for highly flexible sharing relationships, ranging from client-server to peer-to-peer; for sophisticated and precise levels of control over how shared resources are used, including fine-grained and multi- stakeholder access control, delegation, and application of local and global policies; for sharing of varied resources, ranging from programs, files, and data to computers, sensors, and networks; and for diverse usage modes, ranging from single user to multi-user and from performance sensitive to cost-sensitive and hence embracing issues of quality of service, scheduling, co- allocation, and accounting.

Grid computing 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. 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. Grid computing is a service for sharing computer power and data storage capacity over the Internet.

Grid computing uses the Internet to help us share computer power, while the Web uses the Internet to help us share information. Grid computing is making big contributions to scientific research, helping scientists around the world to analyze and store massive amounts of data. A scientist studying proteins logs into a computer and uses an entire network of computers to analyze data. A businessman accesses his company's network through a PDA in order to forecast the future of a particular stock. An Army official accesses and coordinates computer resources on three different military networks to formulate a battle strategy. All of these scenarios have one thing in common: They rely on a concept called grid computing. At its most basic level, grid computing is a computer network in which each computer's resources are shared with every other computer in the system. Processing power, memory and data storage are all community resources that authorized users can tap into and leverage for specific tasks. A grid computing system can be as simple as a collection of similar computers running on the same operating system or as complex as inter-networked systems comprised of every computer platform you can think of. The grid computing concept isn't a new one. It's a

DOI 10.17148/IJARCCE.2015.4841

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

special kind of distributed computing. In distributed computing, different computers within the same network share one or more resources. In the ideal grid computing system, every resource is shared, turning a computer network into a powerful supercomputer. With the right user interface, accessing a grid computing system would look no different than accessing a local machine's resources. Every authorized computer would have access to enormous processing power and storage capacity.

ADVANTAGES OF GRID

The following are the reasons why now we are concentrating on Grids:

- Moore's law improvements in computing produce highly functional end systems.
- The Internet and burgeoning wired and wireless provide universal connectivity.
- Changing modes of working and problem solving emphasize teamwork, computation.
- Network exponentials produce dramatic changes in geometry and geography [17].
- The network potentials are as follows: Network vs. computer performance.
- Computer speed doubles every 18 months.
- Network speed doubles every 9 months.
- Difference = order of magnitude per 5 years.

THE GRID ARCHITECTURE DESCRIPTION

Our goal in describing the Grid architecture is not to provide a complete enumeration of all required protocols (and services, APIs, and SDKs) but rather to identify requirements for general classes of component. The result is an extensible, open architectural structure within which can be placed solutions to key VO requirements. Our architecture and the subsequent discussion organize components into layers, as shown in Figure 1.1. Components within each layer share common characteristics but can build on capabilities and behaviors provided by any lower layer. In specifying the various layers of the Grid architecture, we follow the principles of the hourglass model. The narrow neck of the hourglass defines a small set of core abstractions and protocols (e.g., TCP and HTTP in the Internet), onto which many different high-level behaviors can be mapped (the top of the hourglass), and which themselves can be mapped onto many different underlying technologies (the base of the hourglass).

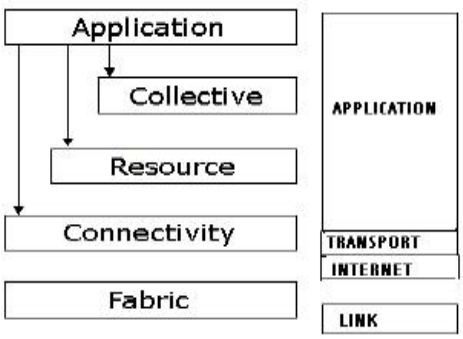

Fig. 1 Architecture and the subsequent discussion organize components into layers

QoS Metrics in a Grid Environment

Latency: This type of metric is related to the time it takes to execute a task and is measured in time units. Latency metrics can be defined at different granularities. For example, in the IC case, one may be interested in the average time it takes to complete immediate quote requests or in the 95-th percentile of the time to respond to non-immediate requests. These two examples are illustrative of QoS metrics at the application layer. Another example of latency metric is the elapsed time to return a delayed quote to the user. Sophisticated RAMs are typically complex parallel jobs that are decomposed into tasks allocated to different computing resources in the grid[3]. An example of a QoS metric at the collective layer is the average time to perform co- reservation and co-allocation of computing resources in order to run a RAM. At the resourcelayer, one may be interested in measuring the time it takes to access a specific computing resource through the GRAM protocol. An example of latency metric at the connectivity layer is the time it takes to perform an authentication on behalf of a customer to a financial organization using the GSI. At the fabric layer, one may be interested in measuring the time taken by a database server at a health insurance organization to reply to a query.

Throughput: It is measured in units of work accomplished per unit time. There are many possible throughput metrics depending on the definition of unit of work. At the application layer one may be interested in the number of delayed quote requests processed per second. At the collective layer one may be interested in the number of queries per second that can be handled by directory services used to locate resources across different VOs. Examples of throughput metrics at the resource layer include i) the effective transfer rate in Kbytes/sec under the Grid-FTP protocol used to transfer files from different computes nodes involved in running RAMs and ii) the number of queries/sec that can be processed by the database server of a law enforcement agency needed by the RAM application. At the connectivity layer, one may be interested in measuring the throughput, in Kbytes/sec, of a secure connection between an instance of a RAM model and a database service that provides financial information about a customer. Finally, an example of a throughput metric at the fabric layer could be the number of CPU cycles per second obtained from a machine involved in processing a task that is part of a parallel RAM evaluation job.

Availability: It is defined as the fraction of time that a resource/application is available for use. In a multilayer context such as the one described for the grid architecture, the notion of availability differs for each layer. At the application layer, one may define availability according to the type of request. For example, the availability of the immediate quote request application is the fraction of time that this application is available. At the collective layer, one may define availability in terms of the services, such as directories and brokering services, needed to locate computing resources to run a RAM instance. Availability

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

at the resource layer may be measured for example as the fraction of time that a specific computing node is available for allocation to a RAM instance. At the connectivity layer, one may define availability as the percentage of time that a proxy authentication request succeeds in authenticating a user request with a law enforcement agency. Finally, an example of availability metric at the fabric layer would be the fraction of time that a computing node is available during the execution of a RAM instance.

II. LITERATURE REVIEW

[1] Sandeep Kaur1, Sukhpreet kaur" Efficient Load Balancing Grouping based Job Scheduling Algorithm in Grid Computing", International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), 2013.

Grid computing is a high performance computing environment to solve large-scale computational demands. Computational grids emerged as a next generation computing platform which is a collection of heterogeneous computing resources connected by a network across dynamic and geographically dispersed organizations, to form a distributed high performance computing infrastructure. In computational grid main emphasis is given on resource management and job scheduling. Job scheduling is a decision process by which application components are assigned to available resources to optimize various performance metrics.

The main goal of scheduling is to maximize the resource utilization and minimize processing time of the jobs. Various research works has been done on job scheduling problem in grid, but still further analysis and research needs to be done to improve the performance of scheduling algorithm in computational grid. Different scheduling algorithms are suitable in different situation based on specific requirement. User jobs might be small and of varying lengths according to their requirements. Certainly, it is a real challenge to design an efficient scheduling strategy to achieve high performance in grid computing. Grouping strategy reduce the communication time of small scale jobs, but allocation of large number of jobs to one resource will increase the processing time and leads unbalancing processing load among the resources in grid computing environment. In this paper, an Efficient Load Balancing Dynamic Grouping based Job Scheduling (ELBDGBJS) approach has been proposed for grouping the fine-grained jobs in grid computing environment. In this scheduling algorithm jobs are scheduled based on number of jobs at particular time and resources capability. Independent fine-grained jobs are grouped together based on the dynamically specified group size and resources characteristics, for all resource utilization and minimize processing time. Hence in this paper, they have specifically focused on improving computational grid performance in terms of equal load balance and total computation time. A comparison of their proposed approach with other existing fine-grained job scheduling strategies is provided. [2]. Sudhir Singh "Performance Optimization in Gang Scheduling In Cloud Computing", IOSR Journal of Computer Engineering (IOSRJCS), 2012.

Cloud computing is a latest new computing paradigm where applications, data and IT services are provided over the Internet. The Job Scheduling is the key role in cloud computing systems. One technique is to use gang scheduling where a set of tasks is scheduled to execute simultaneously on a set of processors. Usually tasks are scheduled by user requirements. So, taking existing model new scheduling strategy proposed to overcome the problems of the performance unpredictability with the help of scheduling technique between user and resources.

[3]. Himani Aggarwal, Er. Shakti Nagpal" Comparative Performance Study of CPU Scheduling Algorithms", International Journal of Advanced Research in Computer Science and Software Engineering, 2014

Some CPU scheduling algorithms such as First-Come-First-Serve (FCFS), Shortest Job First (SJF), Priority, Round Robin (R-R), Multilevel Queue(MQ) and Multilevel Feedback Queue (MFQ has been elaborated and assessed on the basic CPU scheduling objectives i.e.; average waiting(AWT), average turnaround (ATT)and average response time(ART), average CPU utilization (AU)and average throughput(AT). These will form the base parameters in making a decision for the suitability of the given algorithm for a given objective. Comparative performance study of various CPU scheduling algorithm is done in this review paper.

[4]. Neetu Goel "A Comparative Study of CPU Scheduling Algorithms", International Journal of Graphics & Image Processing, 2012

Developing CPU scheduling algorithms and understanding their impact in practice can be difficult and time consuming due to the need to modify and test operating system kernel code and measure the resulting performance on a consistent workload of real applications. As processor is the important resource, CPU scheduling becomes very important in accomplishing the operating system (OS) design goals. The intention should be allowed as many as possible running processes at all time in order to make best use of CPU. This paper presents a state diagram that depicts the comparative study of various scheduling algorithms for a single CPU and shows which algorithm is best for the particular situation. Using this representation, it becomes much easier to understand what is going on inside the system and why a different set of processes is a candidate for the allocation of the CPU at different time.

The objective of the study is to analyze the high efficient CPU scheduler on design of the high quality scheduling algorithms which suits the scheduling goals.

III.PROPOSED METHODOLOGY

The proposed work is about to improve the effectiveness of the grid system along with reliability. In this system we will keep the most frequent data items in cache by estimating the data frequency. As the most required data items are kept in the cache itself, it will improve the hit ratio and improve the reliability of data access.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

The system will give the better service allotment such way that the starvation will not occur over the system.

1. Problem Statement

In the case of computational grids the consumers and providers share their resources and schedule the decision. The work is to increase the success rate of job execution and to minimize the fairness deviation among resources and in this work we propose to achieve both this target simultaneously.

The proposed work is very much inspired from a recommendation system. In this system the new decision making will be taken place respective to delay analysis based thresholding mechanism. The proposed work is about to keep only the most required data items in cache. The requirement depends on the frequency of the reuse of data item. The work is about to identify the frequency of each data items from previous history.

For this analysis the improved inverted list is suggested here. It will also predict the group of data items that should be keep in a sequence to get better optimization. The system will perform the improvement over the best first algorithm along with delay analysis. The system is about to reduce the delay and to minimize the starvation over the process allotment.

2. Objective

1. To Analyses the Existing Grid Resources Allocation Process.
2. To Understand the Concept of the Starvation in Processes Allocation under Grid.
3. To Design an effective algorithm for reduce the time delay.
4. To develop an efficient job scheduling algorithm for resource allocation.
5. To implement the proposed algorithm to see the results outcome.
6. To Generate Results.

The proposed work is about to define a cache data replacement scheme based on frequency analysis. According to this scheme most frequent data items will be kept in cache.

As the most required data items are in cache itself the system will improve the efficiency as well as will improve the hit ratio. The system is presenting an efficient and reliable data replacement scheme in cache memory.

To work with the proposed system we have to simulate the service allotment in grid Market based architectural environment in a programming environment.

Here the work is basically about the improvement over the process allotment. The presented work will show the access to the CPU with access time and the relative parameters.

As the architecture build up the next work is to define any of the existing data replacement algorithm in cache memory. The work is here based on the access of data from internal cache to cpu. The analysis will be done on this existing approach.

IV. RESULTS

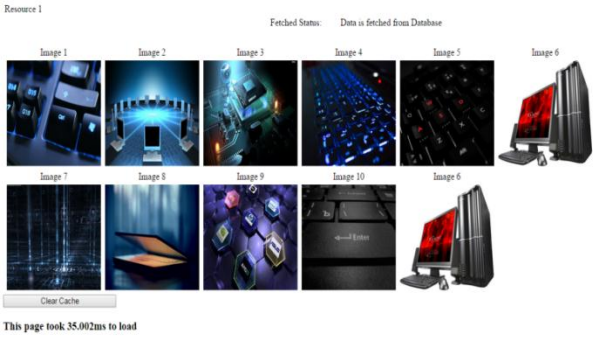

Fig 2 Resource 1 Time without cache

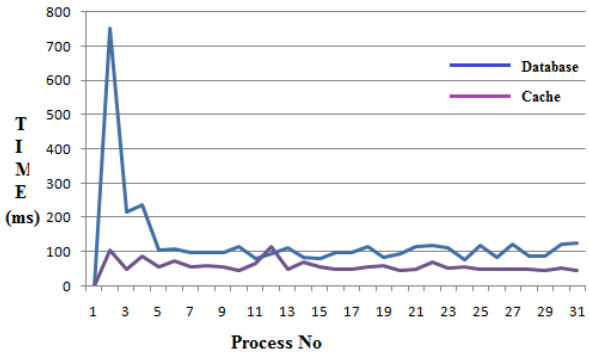

Fig 3 Graphical Analysis

V. CONCLUSION

The improvement of algorithm using cache technique allows us to reduce computational cost because the results calculated in previous times not to be calculated in the progressive process. Besides, Grid computing is also an effective environment to deploy the Multiple Sequence Alignment program. The combination of two techniques brings us many benefits to increase program's execution speed. We have created cache for each grid to reduce waiting time. However, the parallel cache requires large amount of shared storage space, and network bandwidth is also a factor affecting the processing speed of the program. By cache, it won't take long response so that waiting time will also reduce and response time will be increased.

REFERENCES

[1] CCITT Recommendation, "X.509: The Directory – Authentication Framework" 1988.
[2] Chan, S. (2004), "Grid Security at NERSC/LBNL", Globus Security Workshop, 2004.
http://grid.ncsa.uiuc.edu/gw04-security/GW04-SecWkshp-nersc.ppt
[3] Daniel A., Emiliano C. "Quality of Service Aspects and Metrics in GridComputing", 2004
[4] Dierks T. and Allen C., "The TLS Protocol Version 1.0, RFC 2246", IETF, 1999.
[5] Globus S. (2004) CA, http://www.globus.org/security/simple-ca.html, 2004.
[6] H.M. Faheem (2010), "Accelerating Motif Finding Problem using Grid Computing withEnhanced Brute Force".
[7] Haller N., Metz C., Nasser P., and Straw M.(1998), "A One-Time Password System",RFC 2289, IETF, 1998.
[8] Inderpreet C. (2010), "Analyzing the need for Autonomic Behavior in Grid Computing", IEEE.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

[9] M Victor J. (2011), "Object Based Grid Architecture for Enhancing Security in GridComputing", Proceedings of 2011 International Conference on Signal Processing,Communication, Computing and Networking Technologies (ICSCCN 2011) IEEE.
[10] Manoj Kumar M., RakshaS. (2010), "A Memory-Aware Dynamic JobScheduling Model in Grid Computing", 2010 International Conference On ComputerDesign and Applications (ICCDA 2010) IEEE.
[11] Mehdi B. and Ahmad F. (2010), "AGC4ISR, New Software Architecture forAutonomic Grid Computing", 2010 International Conference on Intelligent Systems, Modelling and Simulation IEEE.
[12] NaidilaS. (2011), "Cluster, Grid and Cloud Computing: A Detailed Comparison",The 6th International Conference on Computer Science & Education (ICCSE 2011) August 3-5, 2011. SuperStar Virgo, Singapore.
[13] Peng Zhang Ming Chen Peng-ju He (2010), "The Study of interfacing Wireless SensorNetworks to Grid Computing based on Web Service", 2010 Second InternationalWorkshop on Education Technology and Computer Science, IEEE.
[14] Peter G. (2003), "Plug-and-Play PKI: A PKI your Mother can Use", Use nixSecurity Symposium, 2003.
[15] Syed NasirMehmood S., Ahmad Kamil Bin M. and Alan O. (2010),"Hybrid Resource Allocation Method for Grid Computing", Second InternationalConference on Computer Research and Development, IEEE.
[16] Zhen-chun H., Shi-feng S. (2010), "Design and Implementation of SCO-GADL– a Scientific Computing Oriented Grid Workflow", IEEE 2010.
[17] Krishna Nadiminti , Srikumar Venugopal et al. "The Gridbus Grid Service Broker and Scheduler (v.2.2) User Guide", [Online] Available: http://www.cloudbus.org/broker/2.2/manualv2.2.pdf
[18] Andre Rigland (2010-12-17). "Cloud Computing-How it is Different from Grid Computing". Trial lecturer in conjunction with PhD thesis defence.
[19] Ovaise Khan, "Types of Grid", [Online] Available: http:// thegridweblog.blogspot.in/2005/10/types-of-grids.html.
[20] Luis Ferreira, Fabiano Lucchese, Tomoari Yasuda, Chin Yau Lee, Carlos Alexandre Queiroz, Elton Minetto, Antonio Mungioli First Edition (April 2005). Grid Computing in Research and education.