# The Influence of Various Workload Descriptions on a Grid Scheduling Mechanism

Arkaprava Bhaduri Mandal
Department of Computer Science and Engineering
National Institute of Science and Technology
Berhampur, India
arkabhaduri@gmail.com

Motahar Reza
Department of Mathematics
National Institute of Science and Technology
Berhampur, India
nist_reza@yahoo.com

***Abstract*—Grid Computing is a form of distributed Computing that has emerged as a viable solution to meet the ever increasing needs for computational power and data management capability. Designing solutions in such grid computing framework entails addressing much more complicated issues compared to chore software development, namely concurrency, heterogeneity, scalability and so forth; just to name a few. In order to simplify the task of programming in grid environment a software layer is employed to mask off the massive underlying heterogeneity in network, hardware, operating system and programming languages, known as the middleware. Moreover, resources in a grid are dynamic and thus incorporating appropriate scheduling mechanism becomes a challenging proposition.**

**This paper addresses some major issues in context of job scheduling in computational grids: namely, average active jobs, busy time of CPU, heap memory and average CPU load. They are treated as Work Load Description (WLD) in a grid scenario. For experimentation purposes, Grid Gain has been incorporated as middleware. Experiments have been conducted and subsequent results presented herein demonstrate the efficacy of Grid Gain as a platform of implementation of grid computing for catering to future generation computational needs, including load balancing.**

***Keywords-GridComputing,Scheduling,Work Load Description, GridGain;***

## I. INTRODUCTION

Grid computing ties together unexploited processing cycles of all computers in a network for solving problems too intensive for any stand-alone machine. It is a form of networking unlike conventional networks that focus on communication among devices. It sets aside unemployed CPU capacity in all participating machines to be allocated to one application that is extremely computation intensive and programmed for parallel processing. Grid computing facilitates the virtualization of distributed computing and data 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. An internet user can view an unified instance of content via the Web; a grid user essentially sees a single, large virtual computer [1].

A grid system is formed using many heterogeneous or homogeneous resources to deal with large-scale scientific problems. There are many issues in using grid computing. How to appropriately and efficiently assign resources to tasks, generally called job scheduling, is one of the important issues. The main purpose of job scheduling is to shorten the job completion time and enhance the system throughput. A grid scheduling system should take the various characteristics of grid applications and resources into account [2]. In a grid environment, the resource providers and tasks are all changing constantly, so the traditional scheduling algorithms may not be suitable for a dynamic grid system. It is very important to assign appropriate resources to tasks. Through a good scheduling method, the system can perform better and applications can avoid unnecessary delays.

Grid computing suggests a model for solving massive computational problems by making use of the idle resources (CPU cycles and/or disk storage) of large numbers of disparate computers, often desktop computers, treated as a virtual cluster embedded in a distributed telecommunications infrastructure. Grid computing focus on the ability to support computation across administrative domains sets it apart from traditional computer clusters or traditional distributed computing. Grid computing has the design goal of solving problems too big for any single supercomputer, whilst retaining the flexibility to work on multiple smaller problems. Therefore Grid computing provides a multi-user environment. Its secondary aims are better exploitation of available computing power and catering for the intermittent demands of large computational exercises. The Grid computing infrastructure mainly focuses on the networking services and connections of a large number of computational resources within a grid environment. A Grid is a seamless environment incorporated with the computational and storage capability. The end users can interact with the interface to Grid middleware in order to solve their problems. The basic activities performed by the middleware are resource discovery, scheduling, and the processing of user’s jobs on the globally disseminated Grid resources. The scheduling in Grid emphasizes the problem of mapping the task to the best fit node which is again the essential part of grid computing. The tasks or jobs in the task-set may be independent or dependent to each other. In some Independent task model the size and quantity of tasks are known beforehand where the application comprises some amount of load that may be subdivided into n independent parts. This divisible load theory [3, 4] has already been studied for last few years and applied in a wide variety of different domains [5,6,7,8,9,10,11,12].

In this paper we explore the performance and efficiency of grid with respect to the different workload description. The mean response time or Makespan [13] is the key aspect here in order to evaluate the performance of grid. The experimental setup has been developed using the GridGain middleware. Figure 1 shows the in general architecture of a Grid with geographically distributed resources.

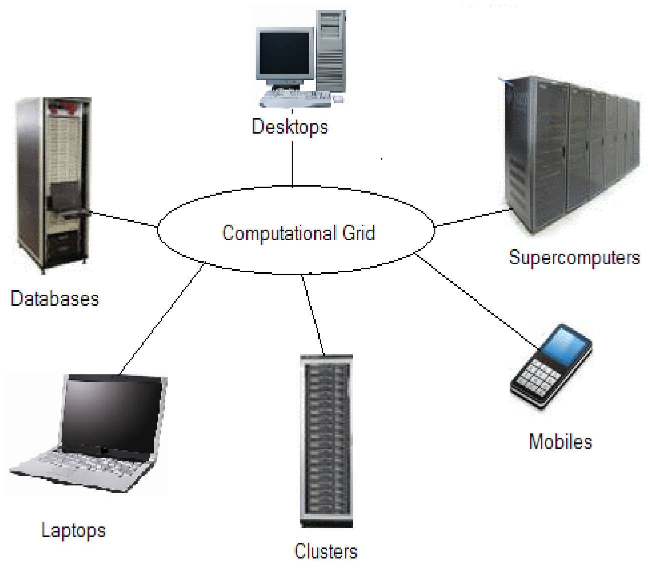

Figure 1. Grid Computing Environment

The rest of the paper is organized as follows. Section 2 deals with the functional and fundamental aspects of Grid Computing. In Section 3 we describe in detail the different methods used to collect the statistical node information along with the entire experimental setup. Experimental results are presented in Section 4 and some conclusion and future works are provided in Section 5.

## II. GRID COMPUTING

Grid computing is an approach which describes a distributed computing infrastructure. It groups together the other technology trends such as Internet, distributed computing and peer-to-peer computing. The explicit focus of grid computing has been identified as synchronized resource sharing and problem solving in dynamic, geographically dispersed virtual organizations. Grid computing involves a growing set of open standards for Web services and interfaces that enable services, or computational resources, accessible over the Internet. Very often grid technologies are used on clusters which can add value on them by supporting with scheduling or making availability of the resources in the cluster. The term grid, and its allied technologies, applies across this entire continuum [14**].**

Grid computing is a form of parallel and distributed computing that involves coordination and sharing of computing, application, data storage and network resources across dynamic and geographically distributed organizations . It is a back-bone infrastructure for web services. Like Internet, which allows sharing of the information, a grid provides sharing of computational power and resources such as data storage, databases, and hardware and software architectures. This integration creates a virtual organization wherein a number of mutually distrustful participants with varying degrees of prior relationship want to share resources to perform some computational tasks [15].

## III. GRID ENVIRONMENT SETUP USING GRIDGAIN

In grid computing environment for Java, GridGain [16] is used as a popular middleware developed in Java for Java developers and is an innate annex of the latest Java development methodologies. The Grid Gain released under the terms of GNU General Public License (GPL) from Grid Gain Systems Inc and used as an open source product. Grid Gain is also suitable for networking systems and applications due to its modern architecture based on Java programming language. It provides a commanding and smart technology to build and run applications on grid computing environment enabling developers to code any custom grid compatible applications or make the sequential one grid compatible and seamlessly deploy it on the grid captivating the fullest benefit of the concepts like affinity load balancing, map-reduce, and peer-to-peer class loading etc. During the deployment of grid using GridGain different working nodes have been created on different machines. There is a management node that is responsible to take the scheduling decisions. A task submitted to a node can be processed locally or can be distributed among the different other working nodes. In this paper mainly the scheduling aspects and the performance of Grid has been analyzed while implementing the grid using GridGain.

The grid set up has been deployed using Grid Gain 2.0.0 as middleware incorporated with 14 nodes. The experiments have been conducted on a set of heterogeneous machines having the following configuration:

Eight individual machines with:

Processor- Intel (R) Core(TM) 2 Duo CPU E7400@2.66GHz., CPU Core Count- 2.Memory- 1024MB, Memory Bus Speed- 800MHz, Hard Drive- 320GB. Ethernet Card. Operating System: Windows (R) XP Professional, SP 2.

Four individual machines with:

Processor- Intel (R) Core (TM) i3-380M Processor (3M Cache, 2.53 GHz), CPU Core Count- 2.Memory- 3GB, Maximum Memory Bandwidth-17.1 GB/s, Hard Drive-500GB. Ethernet Card. Operating System: Windows (R) 7 Home Premium, SP1 64-Bit.

Two individual machines with:

Processor- Intel (R) Core(TM) i5-2430M Processor (3M Cache, 2.40 GHz)., CPU Core Count- 2.Memory- 4GB, Maximum Memory Bandwidth-21.3 GB/s, Hard Drive-500GB. Ethernet Card. Operating System: Windows (R) 7 Home Premium, 64-Bit.

Each node configured with: Middleware: Grid Gain 2.0.0 along with JDK- 6u-10 and Java Runtime Environment with Eclipse 3.2.

All the machines were connected via a switch using RJ-45 cable with a maximum transmission speed of 100.0 Mbps.

For the developers the GridGain provides a lot of flexibility in order to code and deploy their program on grid computing environment. Figure 2 shows the prototype of the experimental setup used for our work

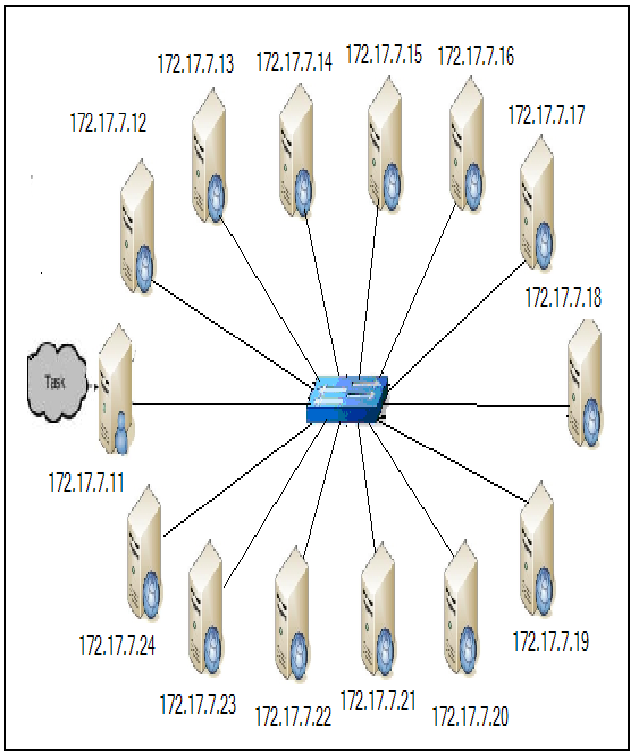

Figure 2. Grid Setup employing GridGain

GridGain has some inbuilt methods by which the various statistical information regarding the various properties of each individual resources can be obtained without any difficulties, like *getAverageCpuLoad(),getAverageActiveJobs(),getAverage CancelledJobs(),getAverageJobWaitTime(),getCurrentCpu Load(),getHeapMemoryUsed(), getHeapMemoryInitialized().*

Initially a simple program of Matrix multiplication has been chosen as the key problem to be used to provide the load on the nodes. A set of task ranging from 50 to 400 is used as load in order to execute the problem on grid.

As the primary objective of this work is to unearth the solution for grid scheduling problem, the working Zone refers to the mapping of the jobs to the fittest nodes. Now the difficulty lies on the selection procedure of the fittest node for a particular job. The Modified Dynamic Fittest Processor Largest Task First (MDFPLTF) algorithm is used to schedule the jobs to the specific nodes. Here the fittest processor has been chosen by the corresponding relative value of the WLD. In our work the various Workload Descriptions have been used as the key aspect to recognize the fittest nodes. The performance of the algorithm has been analyzed with respect to the makespan. The following code is used to get the statistical information of the various grid nodes.

```
if(logic==1){
//Logic 1 is for only considering average CPU load.
   nodeId = nodeIdPassed;
   node = nodePassed;
   parameter =
   node.getMetrics().getAverageCpuLoad()
}
```

The method *node.getMetrics().getAverageCpuLoad* is used to get the average CPU load information of a node. The variable *Logic* is used to select the appropriate WLD. The *nodeId* refers to the corresponding node of which the WLD is acquired.

```
for(int k=0;k<taskSet.length;k++){

   jobs.put(new
   GridJobAdapter<String>(taskSet[k]) {

      execute(){
      return

mappingReduce.deployingInGrid(getArgumen
t());

    },gridList.get(k).node);
 }
}
```

In this above mentioned code the *taskSet* contains the set of jobs submitted to the grid where as the *gridList* contains an ordered list of all the active nodes in grid with respect to the WLD specified. The method *jobs.put()* along with the method *mappingReduce.deployingInGrid(getArgument()* is used here to map the job from the taskSet to the node from the *gridList* as prescribed by the MDFPLTF algorithm.

The experiments have been conducted numerous times for each individual workload descriptions with varying size of task set.

## IV. RESULT AND DISCUSSION

In this paper we investigate the impact of the different WLD in scheduling in order to find out the most suitable WLD using which the proposed algorithm may outperform the inbuilt one as well as provide the better performance as compared to the other WLD. There are several different mechanisms in GridGain using which getting the various statistical information regarding the different heterogeneous resources is very easy.

Here we make a comprehensive study of the performance of different WLD with varying size of taskset. Near about 250 times the experiments have been conducted with various WLD and with varying Task size.

In case of our experiment we choose *Average CPU Load (ACL) ,Heap Memory Committed (HMC), Average Active Jobs (AAJ) and Busy Time Percentage (BTP)* as WLD. Here we study the comprehensive impact of each individual WLD when used with our proposed MDFPLTF algorithm to evaluate the performance and efficiency of the algorithm in grid computing environment

From Figure 3 it can be observed that initially though the basic algorithm outperforms the MDFPLTF with AAJ, but as

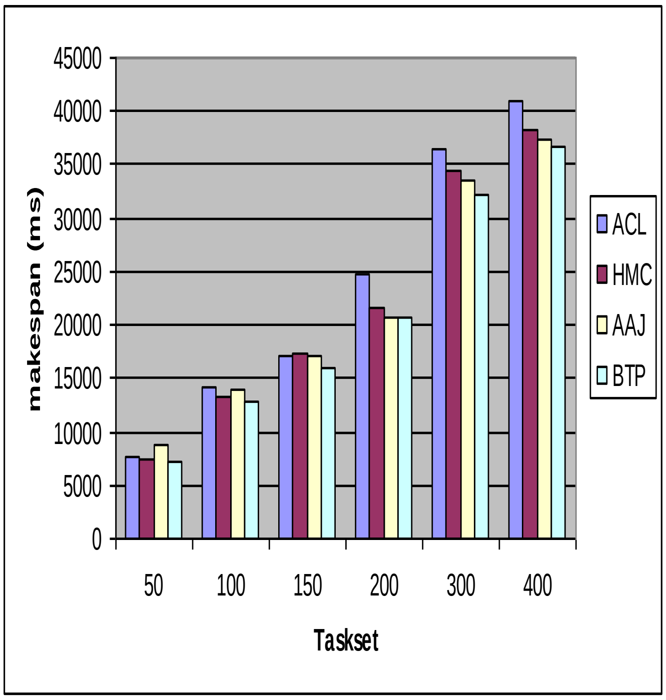

Figure 3. Performance of different workload descriptor for different size of taskset

the load/taskset goes on increasing the AAJ rules over both the MDFPLTF with ACL as well as with HMC, although HMC provides better solution than the ACL. If we compare the efficiency of the different WLD among themselves, it can be found that initially the AAJ gives poor performance as compare to the HMC and BTP. Though Initially the HMC provides better performance but with the increasing taskset it goes down. If we compare the performance curve of AAJ and HMC it is observed that they are converse to each other. Initially the HMC gives better results but as the load goes on increasing it falls where as the AAJ provides poorer solution initially but with increasing load it also increase it's performance. But the optimal solution is provided by the BTP. From the initiation of the experiment to the end it provides a constant best output in comparison with all the other flavours of MDFPLTF algorithm. The different WLD has its own impact on the different versions of the MDFPLTF algorithm.

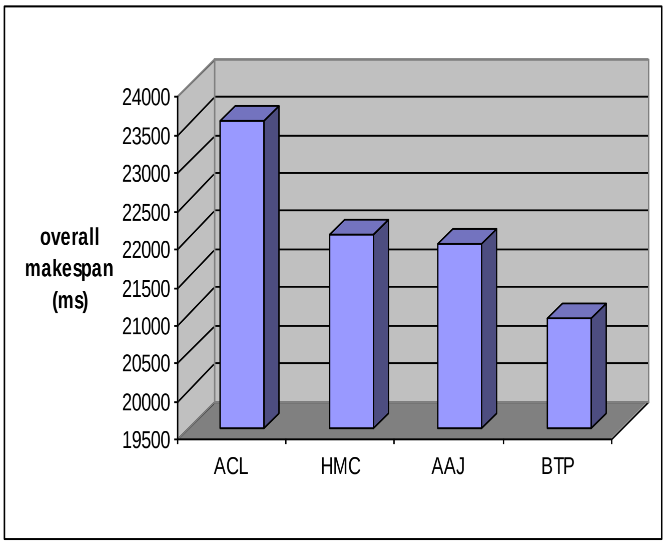

Figure 4. Overall Performance of different workload descriptor

Figure 4 shows the overall performance of the different versions of MDFPLTF algorithm. The empirical result shows that in our experiment in heterogeneous environment, the MDFPLTF with AAJ gives the 2$^{nd}$ best performance whereas the MDFPLTF with BTP provides the optimal solution.

## V. CONCLUSION

In this paper we have reviewed and investigated the various aspects of grid regarding the performance and efficiency. For our experimental setup GridGain is used as middleware. The various workload descriptors such as *busy time percentage* , *average active jobs*, *heap memory committed* as well as the GridGain's own scheduling algorithm are used to evaluate the efficiency and performance of the grid computing environment. Among them the *MDFPLTF with BTP* provides the superior performance in an overall aspect of makespan and tasksize.

The evaluation of efficiency as well as the performance of grid using workload description is always problem specific. In case of our problem of Matrix Multiplication the BTP gives better performance. In future, the various other workload descriptors can be used to assess the performance and efficiency of grid for the different kind of problems.

## REFERENCES

[1] Y. C. Lee, A. Y. Zomaya, "Practical Scheduling of Bag-of-Tasks Applications on Grids with Dynamic Resilience," IEEE Transaction on computers, vol. 56, no. 6, pp. 815-825, 2007.

[2] N.Malarvizhi, V.Rhymend Uthariaraj, "A Minimum Time To Release Job Scheduling Algorithm in Computational Grid Environment", *Proc Fifth Int'l Joint Conf. on INC, IMS and IDC (NCM '09)*,pp. 13-18, 2009.

[3] B. V. Bharadwaj, D. Ghose, V. Mani, and T.G. Robertazzi, SchedulingDivisible Loads in Parallel and Distributed Systems. IEEE CS Press,1996.

[4] Cluster Computing, special issue on divisible load scheduling,vol. 6, no. 1, 2003.

[5] M. Drozdowski and P. Wolniewicz, "Experiments with Scheduling Divisible Tasks in Clusters of Workstations," *Proc. Int'l Conf.Parallel and Distributed Computing (Europar)*, pp. 311-319, 2000.

[6] V. Bharadwaj and S. Ranganath, "Theoretical and ExperimentalStudy on Large Size Image Processing Applications Using Divisible Load Paradigm on Distributed Bus Networks," *Image and Vision Computing*, vol. 20, nos. 13-14, pp. 917-1034, 2002.

[7] C.K. Lee and M. Hamdia, "Parallel Image Processing Applications on a Network of Workstations," *Parallel Computing*, vol. 21, pp. 137-160, 1995.

[8] G. Miller, D.G. Payne, T.N. Phung, H. Siegel, and R. Williams, "Parallel Processing of Spaceborne Imaging Radar Data," *Proc. Int'l Conf. High Performance Computing and Comm. (SC '95)*, 1995.

[9] Y.-J. Chiang, R. Farias, C.T. Silva, and B. Wei, "A Unified Infrastructure for Parallel Out-of-Core Isosurface Extraction and Volume Rendering of Unstructured Grids," *Proc. IEEE Symp. Parallel and Large-Data Visualization and Graphics*, pp. 59-66, 2001.

[10] W. Bethel, B. Tierney, J. lee, D. Gunter, and S. Lau, "Using Highspeed WANs and Network Data Caches to Enable Remote and Distributed Visualization," *Proc. Int'l Conf. High Performance Computing and Comm. (SC '00)*, 2000.

[11] A. Garcia and H.W. Shen, "Parallel Volume Rendering: An Interleaved Parallel Volume Renderer with PC-Clusters," *Proc. Fourth Eurographics Workshop Parallel Graphics and Visualization*, pp. 51-59, 2002.

[12] VisibleHumanProject,2003,http://www.nlm.nih.gov/research/visible/visible_human .html.

[13] Y. Yang, K. Raadt and H. Casanova, "Multiround Algorithms for Scheduling Divisible Loads", *IEEE Transactions on parallel and distributed systems*, vol. 16, no. 11, nov 2005.

[14] M. Ali, Z.Y. Dong, P. Zhang, "Adoptability of grid computing technology in power systems analysis, operations and control", *IET Generation, Transmission & Distribution*, Vol. 3, Issue 10, pp. 949–959, 2009.

[15] I. Foster, C. Kesselman. **"**Computational Grids**".** *Chapter 2 of "The Grid: Blueprint for a New Computing Infrastructure"*, Morgan-Kaufman, 1999.

[16] GridGainMiddleware. http://www.gridgain.com/online_resources .html