Load Balancing Dynamic Job Grouping Based Schedulimg Algorithm in Grid Computing

Sandeep Kaur, Sukhpreet Kaur

Abstract— Grid Computing is a high performance computing that solves complicated tasks and provides powerful computing abilities. The main purpose of Grid computing is to share the computational power, storage memory, network resource to solve a large problem. Efficient Resource management and job scheduling algorithm are key issue in grid computing. 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. In our previous work we surveyed various grouping based job scheduling algorithm and found that 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 we proposed a "Load balancing Dynamic Job Grouping-Based Scheduling in Grid Computing" with the objective of balance the processing load among the selected resources in grid computing environment at some extent and reducing overall processing time of jobs and improves the throughput of grid resources.

Index Terms—Algorithms,Fine-grained job, Grid computing,Grouping strategy,Job scheduling , Load balancing, Resources management.

1 INTRODUCTION

THE Grid" takes its name from an analogy with the electrical "power grid. The idea was that accessing computer power from a computer grid would be as simple as accessing electrical power from an electrical grid "The term Grid computing originated in the early 1990 as a metaphor for making computer power as easy to access as an electric power. The ideas of the grid (including those from distributed computing, object oriented programming, cluster computing, web services and others) were brought together by Ian Foster, Carl Kesselman and Steve Tuecke, widely regarded as the "fathers of the grid". Grid Computing is a form of distributed computing based on the dynamic sharing of resources between participants, organizations and companies to by combining them, and thereby carrying out intensive computing applications or processing very large amounts of data. Such applications would not be possible within a single body or company.

There are a wide range of heterogeneous and geographically distributed resources in grid, such as computational resource, storage resource, equipment resource, and so forth. Grid resources are geographically distributed across multiple administrative domains and owned by different organizations. Resource management and application scheduling are very complex due to the large-scale heterogeneity presented in resources, management policies, users, and applications requirements in these environments. Consequently, many researches are motivated in different aspects of the job scheduling algorithm. One motivation of grid computing is to aggregate the power of widely distributed resources, and provide non-trivial services to users [4]. But, there exist several applications with a large number of lightweight jobs. It is wasteful to process a lightweight job with a highly capable computer. The total overhead of fine-grained jobs scheduling can be reduced by grouping the lightweight jobs during the scheduling process for deployment over the grid resources. Various job grouping based scheduling algorithms in grid computing have been developed.

Grouping strategy plays a vital role to improve the overall performance of grid computing environment by minimizing the communication time. But in existing job grouping based scheduling strategies, grouping is totally based on resources capability and bandwidth rate. Jobs are added into the group until selected resource is not fully occupied. In this way first resources in sorted out resource list are highly loaded where as last resources might be lightly loaded and idle. Processing load among the selected resources are not balanced. Allocation of large number of jobs to one resource will increase the processing time [7] and leads unbalancing processing load among the resources in grid computing environment [2].

Load Balancing is a technique which is used to distribute the workload equally across multiple computers to enhance resource utilization and to reduce the response time in grid environment. Main goal of load balancing is to balance the load across all the processors. It improves the throughput of grid resources. A good Scheduling algorithm should assign jobs to resources efficiently and balance the system load [8]. The proposed job scheduling strategy takes into account: (1) number of jobs and resources (2) the processing requirements for each job, (3) the grouping mechanism of these jobs, known as a job grouping, according to the number of jobs in each group, and (4) the transmitting of the job grouping to the appropriate resource.

The job grouping is done based on a number of resources and the total number of jobs.

.

2 RELATED WORK

Although Grids have been used extensively for executing in-

tensive jobs, but there are also exist several applications with a large number of lightweight jobs which needs small processing requirements. Sending/receiving each small job individually to/from the resources will increase the total communication time and cost. These applications involve high overhead time and cost in terms of job transmission to and from Grid resources and, job processing at the Grid resources. [3] Grouping strategy efficiently reduces the processing time of jobs in comparison to others. The total processing capabilities of each resource may not be fully utilized as; each time the resource receives a small scaled job. Job grouping strategy aims to fully utilize the resource reduce the drawbacks on the total processing time and cost Grid is type of parallel and distributed system that enables the sharing, selection and aggregation of geographically distributed resources dynamically at run time depending on their availability, capability, performance, cost, and user quality-of self-service requirement[4] A dynamic job grouping-based scheduling algorithm in [5], jobs are grouped according to MIPS of the resource. This model reduces the processing time and communication time of job, but the algorithm doesn't take the dynamic resource characteristics into account and the grouping strategy can't utilize resource sufficiently. Bandwidth-Aware Job Grouping-Based scheduling strategy that schedules the jobs according to MIPS and bandwidth of the resource and the model sends grouped jobs to the resource whose network bandwidth has highest communication or transmission rate [6]. The TMDGJS in [7] terms of processing time and overhead time, gives better performance, by minimizing overhead time and computation time. This strategy play important role in reducing overall processing time of jobs. Jobs are put alternatively from shortest list and added into job group according to the processing capability of the selected resource. However, in [8] the author state that allocating large number of jobs to one resource will increase the processing time. So to avoid this situation during job grouping activity, the total number of jobs group should be created such that the processing loads among the selected resource are balanced. The term "load balancing" [1] refers to the technique that tries to distribute work load between several computers, network links, CPUs, hard drives, or other resources, in order to get optimal resource utilization, throughput, or response. To minimize the time needed to perform all tasks, the workload has to be evenly distributed over all nodes which are based on their processing capabilities. This is why load balancing is needed. The load balancing problem is closely related to scheduling and resource allocation. Grouping strategy based on the processing capability of selected resource. After resource selection, Jobs are started to put into the group. This process continues until the resource capability is less to the sum of grouped job requirement. This technique is also useful in maximum resource utilization, as jobs are grouped based on resource capability, but some system are fully occupied whereas other might be remains unutilized or idle in grid environment [2].

3 GROUPING STRAEGY

Jobs are received and stored in array.Then resources are sorted out in descending order of their processing capabilities in MIPS and bandwidth in Mb/s. and jobs according to the job length .After that total number of jobs and resoures are calculated and number of jobs in each group is speciefied .

$$G\_size = JList\_Size / RList\_Size$$

jobs are added into group based on group size by alternatively taking jobs from front end i.e. job with higher length and then rear end of the job list i.e. job with smaller length While grouping the jobs taken in order of the grouping strategy from the job list, the above strategy falls into cases given below.

$$G(i)\_size <= G\_size$$

When above condition fails while adding a job alternatively from job list during the grouping operation, then it stops grouping and sends the job group to the dispatcher. Then, it takes resource from sorted out resources list and job group sent to the corresponding resource for computation. The above job grouping process is repeated as long as jobs exist.

The adventage of this grouping strategy is as follows.
1) Balance the processing among the selected resources.
2) To start the grouping process with the addition of a job that involves bigger computational need.
3) To ensure a well combination of bigger and smaller jobs added into the group

The proposed job grouping and scheduling algorithm presented next is illustrated through an example. In this example 15 user jobs with varying processing requirements (MI) are grouped into five job groups taking into account the number of jobs group and according to the processing capabilities (MIPS) of the available resources and the Chunk_size. Resource MI is calculated as: RMI=MIPS*Chunk_Size and the Chunk_Size is taken as 10. Before grouping, the group_size is speciefied. The jobs are added into according to the grouping strategy as mentioned and (the first job is always taken from the front end of the reverse sorted job list Then, the next job is added from the rear end of the list).

4 ALGORITHM

1. The scheduler receives the job_List, j[ ] created by the user.

2. Sort the job_List, according to their MI in decreasing order.

3 .Count the jobs and store in variable JList_Size .

4. The scheduler receives the Resource_List, R [j].

5. Count the number of resources and store in RList_Size variable.

6 Sort the Resources in decreasing order according to their MIPS.

7. Get the MIPS of the Resource j.

8. Resource_MI=R[j] x Chunk_Size.

9. Define the total size of each group

TG_size = JList_Size / RList_Size

10. Set the Group(i)_size to zero. (i=0,1,2,3……n)

11. Set the Grouped_Gridlet_Length to zero.

If Group(i)_size<=TG_size and

Grouped_Gridlet_Length is less than Resource[j]_MI

job are selected from sorted out job_list alternatively and

added to summation of previous Grouped_Gridlet_Length and

Group(i)_size is increment

else

Deduct the last job added to Group(i)_size

12. Submit the Group(i)_size to Resource, R[j].

13 Increment i (for next group)

14 Increment j (for next resource in Resource-List) and go-to step 10

The proposed job grouping and scheduling algorithm presented next is illustrated through an example. In this example 15 user jobs with varying processing requirements (MI) are grouped into five job groups taking into account the number of jobs group and according to the processing capabilities (MIPS) of the available resources and the Chunk_size. Resource MI is calculated as: RMI=MIPS*Chunk_Size and the Chunk_Size is taken as 10. Before grouping, the group_size is specified. The jobs are added into according to the grouping strategy as mentioned and (the first job is always taken from the front end of the reverse sorted job list Then, the next job is added from the rear end of the list.

Fig 1 presents a percentage of processing load using the both TMDGJS strategy and LBDGJS strategy.

5 EXAMPLE

Job_list with job length(MI)

1	2	3	4	5	6	7
114	95	30	10	120	35	91

8	9	10	11	12	13	14	15
50	66	150	85	15	5	2	8

R_list with MIPS

Re-sourceID	R1	R2	R3	R4	R5
MIPS	20	10	30	15	25

Chunk Size=10

RMI=MIPS*chunk size

Group No.	Resource id	Resource RMI	TMDGJS	LBDGJS
1	3	300	98%	91%
2	5	250	90%	86%
3	1	200	78%	92%
4	4	150	90%	83%
5	2	100	65%	80%

Fig.1

Fig. 1 Comparison of TMDGJS and LBDGJS based on processing load among the resources

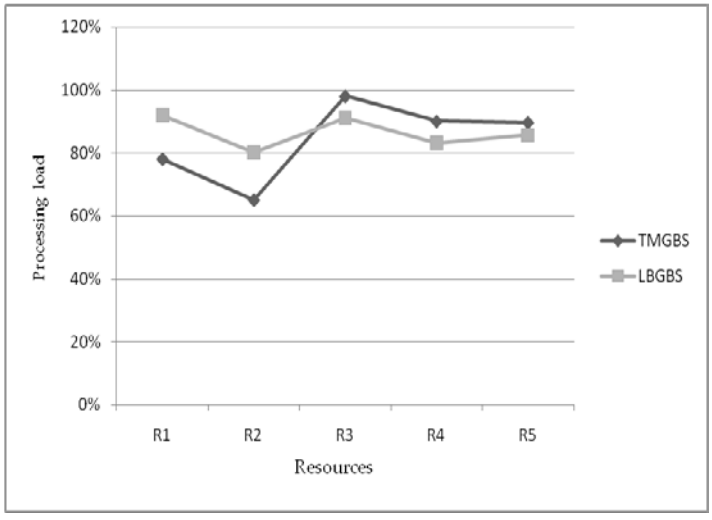

Fig.2

Fig 2 depicts the processing load comparison of two algorithms

for processing numbers of jobs with different MI on five numbers of resources

MI	: Million instructions or processing requirements of a user job
MIPS	: Million instructions per second or processing capabilities of a resource
Processing Time	: Total time taken for executing the user jobs on the Grid
Job_List	: List of user jobs
RList	: List of available Grid resources
JList_Size	: Total number of user jobs
RList_Size	: Total number of available Grid resources
Chunk size	: (time in seconds) for the job grouping activity
RMI	: MIPS*chunk size
G_size	: Total size of each group
G(i)_size	: size of each group
Grouped_Gridlet_Length	: Length of jobs in million instructions in group.

Used terms and their deffinition

6. CONCLUSION

The job grouping strategy results in increased performance in terms of balance the processing load among the resources if it is applied to a Grid application with a large number of jobs where each user job holds small processing requirements. The strategy groups the small scaled user jobs into few job groups. This reduces the communication overhead time and processing overhead time of each user job. The Results show that the proposed LBDGJS is better than TMDGJS for balance the processing load among the resource which improves the throughput of resources.

REFERENCES

[1] Ashish Revar, Malay Andhariya, Dharmendra Sutariya,*Load Balancing in Grid Environment using Machine Learning – Innovative*

[2] Sandeep Kaur, Sukhpreet Kaur, *Survey of Resource and Grouping Based Job Scheduling Algorithm in Grid Computing". International Journal of Computer Science and Mobile Computing. IJCSMC, Vol. 2, Issue. 5, May 2013, pg.214 – 218*

[3] Nithiapidary Muthuvelu, Junyang Liu, Nay Lin Soe, Srikumar Venugopal, Anthony Sulistio and Rajkumar Buyya, *A Dynamic Job Grouping- Based Scheduling for Deploying Applications with Fine-Grained Tasks on Global Grids.*

[4] Quan Liu Yeqing Liao, *Grouping-Based Fine- grained Job Scheduling in Grid Computing" Vol.1,pp.556- 559. IEEE First International Workshop on Education Technology and Computer Science, 2009.*

[5] T.F. Ang, W.K. Ng, T.C. Ling, L.Y. Por and C.S. Liew, 2009. *A Bandwidth -Aware Job Grouping -Based Scheduling on Grid Environment. Information Technology Journal, 8: 372-377*

[6] Manoj Kumar Mishra, Prithviraj Mohanty, G. B. Mund, *A Time-minimization Dynamic Job Grouping based Scheduling in Grid Computing", International Journal of Computer Applications (0975 – 8887) Volume 40– No.16, February 2012*

[7] Pinky Rosemarry, Ravinder Singh, Payal Singhal And Dilip Sisodia, *Grouping Based Job Scheduling Algorithm Using Priority Queue And Hybrid Algorithm in Grid Computing.*

[8] R. Manimala , P.Suresh, *Load balanced job scheduling approach for grid environmentVolume 1, Issue No. 2, April 2013.International Journal Of Advanced Research In Computer Science And Applications. Volume 1, Issue No. 2, April 2013*