Fine grained group based job scheduling algorithm with highest computing resource in grid

Achyut Sakadasariya[1#]
[1] Shantilal shah engineering college Bhavnagar, Gujarat, India.
#Achyut.patel13@gmail.com

Abstract— *Load balancing is the process of load distribution, handling incoming requests and better resource utilization. In a distributed grid computing system it is desirable to achieve an efficient distribution of workload among systems so that each and every machine would have the same workload. No machine should remain idle while other machines are overloaded. Load distribution is done to achieve better response time, better resource utilization and thus improved performance. For improve the performance we have various load balancing algorithms, different types of load balancing strategies and techniques.*

Keywords- Computational grid, resource utilization, request handling, data migration, fine grained, job scheduling.

I. INTRODUCTION

Generally there are three type of phases related to Load balancing i.e. Information Collection, Decision Making, Data Migration. Grid computing is a type of parallel and distributed system that enables the distribution, selection and aggregation of geologically resources dynamically at run time depending on their availability, capability, performance, cost, user quality-of–self-service requirement [1]. Grid Computing should enable the job in question to be run on an idle machine elsewhere on the network [2]. Grids functionally bring together globally distributed computers and information systems for creating a universal source of computing power and information [3].

A key characteristic of Grids is that resources (e.g., CPU cycles and network capacities) are shared among various applications. Load balancing is a technique to enhance resources, utilizing parallelism, exploiting throughput improvisation, and to reduce response time through an appropriate distribution of the application [4].

A key characteristics of grid is that all the resources are shared among various applications and therefore, the amount of resources available to any given application highly fluctuates over times. Load balancing is the technique to enhance resources, utilizing Parallelism, exploiting throughput improvisation, and to cut response time through the appropriate distribution of the application. To minimize the decision time is one of the objectives for load balancing which has yet not been achieved.

However, grid performance can be improved in terms of job processing time by making sure all the resource available in the grid are utilized optimally [4] using a good job scheduling algorithm. In traditional computing system, job scheduling is well studied problem. Job scheduler are exists for many computing environments. These scheduler systems are designed to work under the assumption that they have complete control of a resource and thus can implement the mechanisms and policies needed for the effective use of that resource. Unfortunately, this assumption does not apply to the grid. When dealing with the grid we must develop methods for managing grid resources across separately administered domains, with the resource heterogeneity (grid resources are typically heterogeneous) and differences in local policy.

II. RELATED WORK

This section presents a brief expansion of the work already done in this field so I am trying to improving efficiency of resource utilization.

The scheduler select the available highest priority resource and groups independent lightweight jobs together based on chosen resources processing capability. These jobs are grouped in such a way to maximize the utilization of resources. After grouping all jobs sends to the corresponding resources. This algorithm minimize the network latency, but the scheduling strategy is not ensuring that the resource having a sufficient bandwidth to send the group jobs within required time.

In [5], T.F. Ang, *et. al.* presents a grouping based bandwidth-aware scheduling strategy that schedules the jobs by taking into consideration of their computational capabilities and the communication capabilities of the resources, same as [6]. But the job groups are sent to the corresponding resources based on Largest Job First (LJF) strategy and the results of the processing are sent back to the user after they have been computed at their respective resources. The principle behind the bandwidth-aware scheduling is the scheduling priorities taking into consideration the communication capabilities of the resources. This does not considered the dynamic resource characteristics into account and the scheduling strategy is not ensuring that the resource having a sufficient bandwidth to send the group jobs within required time same as above approach.

A grouping-based fine-grained job scheduling model is presented in [7] by Liu and Liao, the fine-grained jobs grouped into forming coarse-grained are allocated to the available resources according to their processing capabilities and network bandwidth in Largest Job First(LJF) order. But here resource are selected in FCFS order, there is no priority for selecting resources.

- Job Scheduling Model

The four basic building blocks of grid model are user, job scheduler, Grid Information System (GIS) and resources. User jobs submitted to the grid scheduler for scheduling to the resources with an objective of minimizing the processing time and utilizing the resources effectively. The scheduling framework depicts the design of the job scheduler and its interactions with other entities. The job scheduler is a service that resides in a user machine. When the user creates a list of jobs in the user machine, these jobs are sent to the job scheduler for scheduling. The job scheduler obtains information of available resources from the Grid Information Service (GIS) [8]. Based on this information, the job scheduling algorithm is used to grouping the jobs and then resource selection for grouped jobs. When all the jobs are put into groups with selected resources, the grouped jobs are dispatched to their corresponding resources for computation by the dispatcher [9].

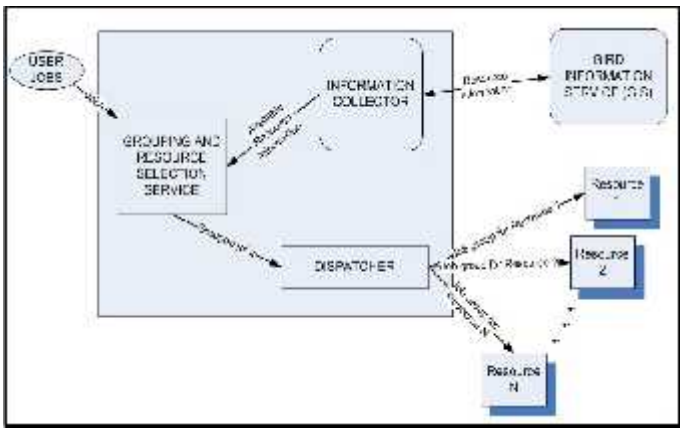

Figure 1. Scheduler model for job grouping

III. PROPOSED WORK

- Algorithm Description

In this section we present a job grouping-based fine-grained job scheduling algorithm. The algorithm mainly consists of two phases: (1) Create available resource list and (2) Job grouping and scheduling. The jobs are described as: *Job = {JOB IDi, JOB MIi, JOB INPUT SIZEi}* Resources are described as: RES_j = {*RES IDj, RES MIPSj, RES BW RATEj, RES COSTj*} after grouping job new Group jobs will be created, described as:

Grouped JOBk = {GroupedJOB IDk, GroupedJOB MIk, Grouped JOB INPUT SIZEk}.

- ***Algorithm1:*** **Listing of the Available Resources**

1: Create a List *RES LIST*.

2: Collect all available resource characteristics and add them to *RES LIST*.

3: Sort the *RES LIST* in descending order with respect to available processing capability (in MIPS).

- ***Algorithm 2:*** **Job Grouping and Scheduling**

1: **while** *JOBLIST.SIZE!* = 0 **do**
2: **for** *i* = 0→*JOBLIST.SIZE*− 1 **do**
3: **for** *j* = 0→*RES LIST.SIZE*− 1 **do**
4: **if** (((*GroupedJOBMI*$_j$ + *JOBMI*$_i$) < (*RESMIPS*$_j$ ∗ *Graularity Tim*
and (((*GroupedJOBINPUT SIZE*$_j$ +*JOBINPUT SIZE*$_i$)/*RESBWRA*
)< *GraularityTime*)) **then**
5: assign the JOB i to GroupedJOB j;
6: update *JOB*$_i$ status to assigned with*RESID*$_j$;
7: break;
8: **end if**
9: **end for**
10: **if** *JOB*$_i$ is not assigned **then**
11: Add the *JOB*$_i$ to*JOBLIST*$_1$;
12: **end if**
13: **end for**
14: **for***k* = 0→*RES LIST.SIZE*− 1 **do**
15: **if** Grouped JOB k is not empty **then**
16: Create a new job with all the Grouped Job assigned to *RES*
17: Assign a unique *Grouped JOBID* to newly created *Grouped JOB LIST*;
18: Allocate the *Grouped JOB*$_k$ to *RES*$_k$;
19: **end if**
20: **end for**
21: **if***JOBLIST*$_1$*.SIZE* != 0 **then**
22: *JOBLIST* = *JOBLIST*$_1$;
23: Update resource information using algorithm 1 ;
24: **end if**
25: **end while**
26: **for** *i* = 0→*GroupedJOBLIST.SIZE*− 1 **do**
27: Receive computed *Grouped JOB*$_i$ from respective resources;
28: **end for**

IV. CONCLUSIONS AND FUTURE WORK

We have discussed about the problem of job scheduling in heterecious computational grid, where user submits jobs with a large number of lightweight jobs (requires small processing requirement) and we have tried to find a solution for that problem. We have proposed an efficient Dynamic Bandwidth-Aware Job-Grouping Based Scheduling algorithm. We have got some better performance in terms of processing time and processing cost than some of the job existing grouping based schemes. In the simulation results, it is clear that proposed approach reduces the total processing time and processing cost. However, allocating a large number of Gridlets to one resource will increase the total processing time and processing cost. Therefore, to avoid this situation during job grouping activity, the total number of Gridlet group should be created such that the processing load among the

selected resources are balanced. The proposed approaches have been critically analyzed and few limitations have been observed. These limitations can be studied and refined. Additionally, the simulation environment is semi-dynamic and it can't reflects in the real computational grid environment sufficiently that promotes further research in the proposed area. In future research some more factors like current load of the resource, jobs with a deadline, network delay, [10] QoS (Quality of Service) [11] requirements will be taken into account. The proposed approaches have been critically analyzed and few limitations have been observed. These limitations can be studied and refined. Additionally, the simulation environment is semi-dynamic and it can't reflects in the real computational voice AC respectively. Our proposed technique attains high delivery ratio and throughput with reduced delay when compared with the existing technique

REFERENCES

[1] *Computer*, 35:37–46, June 2002.

[2] Klaus Krauter, Rajkumar Buyya, and Muthucumaru Maheswaran. A taxonomy and survey of grid resource management systems for distributed computing. *Softw. Pract. Exper.*, 32:135–164 , February 2002.

[3] Ian Foster and Carl Kesselman. Grid resource management. chapter The Grid in a nutshell, pages 3–13. Kluwer Academic Publishers, Norwell, MA, USA, 2004.

[4] Rajkumar Buyya and Srikumar Venugopal. A gentle introduction to grid computing and technologies. *CSI Communications*, 29(1):9–19, July 2005. Computer Society of India (CSI).

[5] Ann Chervenak, Ian Foster, Carl Kesselman, Charles Salisbury, and Steven Tuecke. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *JOURNAL OF NETWORK AND COMPUTER APPLICATIONS*, 23:187–200, 1999.

[6] Fatos Xhafa and Ajith Abraham. Computational models and heuristic methods for grid scheduling problems. *Future Generation Computer Systems*, 26(4):608 – 621, 2010.

[7] Jarek Nabrzyski, Jennifer M. Schopf, and Jan Weglarz, editors. *Grid resource management: state of the art and future trends*. Kluwer Academic Publishers, Norwell, MA, USA, 2004.

[8] 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. In *Proceedings of the 2005 Australasian workshop on Grid computing and e-research - Volume 44*, ACSW Frontiers '05, pages 41–48, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[9] Ng Wai Keat, Ang Tan Fong, Ling Teck Chaw, and Liew Chee Sun. Scheduling framework for bandwidth-aware job grouping-based scheduling in grid computing. *Malaysian Journal of Computer Science*, 19(2):117–126, 2006.

[10] Quan Liu and Yeqing Liao. Grouping-based fine-grained job scheduling in grid computing. In *Proceedings of the 2009 First International Workshop on Education Technology and Computer Science - Volume 01*, pages 556–559, Washington, DC, USA, 2009. IEEE Computer Society.

[11] GridSim 5.0 Beta Application Programming Interface(API). http://www.buyya.com/gridsim/doc/api/.