International Journal of Computer Science and Business Informatics

IJCSBI.ORG

Comparative Analysis of Job Scheduling for Grid Environment

Neeraj Pandey
Department of Computer Science & Engineering
G. B. Pant Engineering College Ghurdauri
Uttarakhand, India.

Ashish Arya
Department of Computer Science & Engineering
G. B. Pant Engineering College Ghurdauri
Uttarakhand, India

Nitin Kumar Agrawal
Department of Computer Science & Engineering
G. B. Pant Engineering College Ghurdauri
Uttarakhand, India

ABSTRACT
Grid computing is a continuous growing technology that alleviates the executions of large-scale resource intensive applications on geographically distributed computing resources. For a computational grid environment, there are number of scheduling policies available to address the scheduling and load balancing problem. Scheduling techniques applied in grid systems are primarily based on the concept of queuing systems, and deals with the allocation of job to computing node. The scheduler, that schedules the incoming job can be based on global vs. local i.e. what information will be used to make a load balancing decision, centralized vs. de-centralized i.e. where load balancing decisions are made, and static vs. dynamic i.e. when the distribution of load is made. The primary objective of all load balancing algorithm is minimization of the makespan value, maximum load balanced and to gain more desirable performance. In this paper, we present the various load balancing strategies of job scheduling for grid computing environment. We also analyze the efficiency and limitations of the various approaches

Keywords
Computing, Load Balancing, Scheduling, Genetic Algorithm, Fuzzy logic, Job Replication.

1. INTRODUCTION
In the last several years grid computing has emerged as an weighty field, distinguished from conventional distributed computing by its focus on large-scale resource sharing, innovative applications, and in some cases, high-performance orientation [1]. These enable sharing, selection and aggregation of suitable computational and data resources for solving large-scale data intensive problems in science, engineering, and commerce [5]. A grid computing environment comprises combination of various homogeneous and heterogeneous resources such as computing nodes, and workstations

which are virtually aggregated to serve as a unified computing resource. Grid middleware provide users with seamless computing ability and uniform access to resources in the heterogeneous grid environment. In order to provide user with a seamless computing environment, the Grid middleware system need to solve several challenges originating from the inherent features of the grid [2].

In distributed systems, every node has different processing speed and system resources, so in order to enhance the utilization of each node and shorten the consumption of time, "Load Balancing" will play a critical role [15]. The performance of load balancing algorithms is strongly related to the number of computing node. Since each computing node has its own inimitable computing capabilities and the pattern of the job arrival to the computing node is imbalanced, thus the grid system may be overloaded. The main objective of load balancing is improved the performance of grid system through its distribution of load among the computing nodes, and minimize the execution time of job. In general, load-balancing algorithms can be categorized as centralized or decentralized in terms of where the load balancing decisions are made. A centralized load balancing approach can function either based on averages scheme or instantaneous scheme according to the type of information on which the load balancing decisions are made [4].

The rest of paper is organized into 6 sections. Section 2 presents the overview of the system model including the grid and mathematical model with load balancing architecture Section. The load balancing approaches for grid system are presented in Section 3. The analysis and comparisons of some grid load balancing algorithms are described in section 4. Section 5 presents some challenges and key issues related to load balancing And finally, the conclusion is given in section 6.

2. SYSTEM MODEL

2.1 Grid Model

The grid under study consist a central resource management unit (RMU), to which every computing node (CN) connects and grid clients send their job to RMU for further processing. The RMU is responsible for scheduling jobs among CN. The role of dispatcher is the job management, including maintenance of the load balancing, monitoring node status, node selection, execution, and assignment of jobs for each node. An agent provides a simple and uniform abstraction of the functions in the grid management. The CNs in the grid can be either homogeneous or heterogeneous and a queue is associated with every computing node. The arrived jobs will be placed in a job queue in the RMU from which they are assigned to CNs. The grid agent monitors the waiting job. Upon arrival, jobs must be assigned

either to exactly one CN for processing immediately by the instantaneous scheduling or wait to be scheduled by the averages-based scheme [4]. In a Grid system, the composition of nodes is dynamic, every node is likely to enter a busy state at any time and thus lower its performance, so in selecting nodes, all factors should be considered. At the global grid level, each agent is a representative of a grid resource and acts as a service provider of high performance computing power [14].

2.2 Mathematical Model

A computational grid system model [21] consisting p set of the CNs, is shown in figure 1. It consists of Ni computing nodes (CNs) such that:

$$N_i = \{ (N_1, N_2, N_3, \ldots\ldots\ldots\ldots, N_{n-1}, N_n), \ |N| \ = n \} \quad (1)$$

The nodes are connected to each other via a communication network. The nodes of grid system possibly could be either an individual machine or a cluster. The nature of the node is combination of both homogeneous and heterogeneous, and modeled as M/M/1 queuing system. The inter arrival time and service time of the system follows exponentially distributed. The notations and assumptions used are as follows:

- $\emptyset_i$: External job arrival rate at node i.
- μ_i : Mean service rate of node i.
- λ : Total traffic through the network.

A job arriving at node i may be either processed at node i or transferred to another node j through the communication network for computation. The performance of load balancing policies are closely related to the number of node involves in a computational grid system.

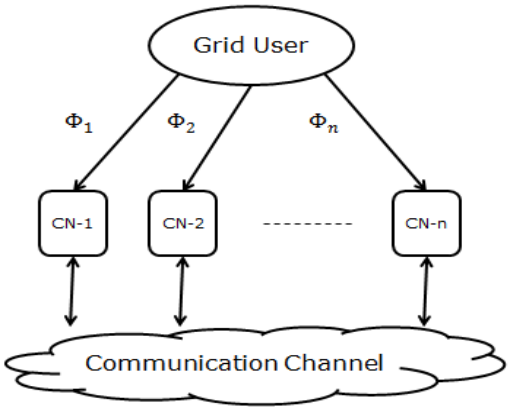

Figure 1. Grid System Model

- Φ : Total external job arrival rate of the system and given as:

$$\Phi = \sum_{i=1}^{n} \phi_i \tag{2}$$

- r_i : Mean service time of a job at node i (i.e. the average time to service (process) a job at node i).
- β_i : Mean job processing rate (load) at node i (i.e. the average number of jobs processed at node i per unit interval of time). This is the load for node i assigned by the job allocation scheme.
- t : Mean communication time for sending or/and receiving a job from one node to another node.
- ρ : Utilization of the communication network and given as:

$$\rho = t * \lambda \tag{3}$$

3. GRID LOAD BALANCING APPROACH

In this section, four algorithms are considered for grid load balancing. A grid is a huge collection of multiple grid resources (local or global) that are distributed geographically in a wide area. Load balancing is an important system function destined to distribute the workload among available computing nodes to improve throughput and/or execution times of parallel computer programs either uniform or non-uniformly [19]. There are various load balancing algorithms are projected in past several years. Various approaches such as fuzzy logic, genetic algorithm, and job replication are used to implement load balancing algorithms.

3.1 FCFS Approach

The FCFS algorithm [4,14] is proved to be efficient under some conditions. Consider a grid environment G with n CNs as given in equation 1, where each CNs Ni has its own capability to process the job. Let m be the total number of job J, which is considered to be run on G.

$$J_i = \{J_1, J_2, J_3, ..., J_{m-1}, J_m\} \tag{4}$$

The arrival time of each job Jj is tj and execution time is txj. Each job has two scheduled attributes: a start time and an end time denoted by ts, te, respectively. Upon arrival, jobs are allocated to a certain CN Nx ∈ N by the central resource management unit using first-come-first-served algorithm. The function of the agent is to find the earliest possible time for each job to complete, according to the sequence of the job arrivals. A job probably allocated to any of the CN. So, the function of FCFS to consider all these possibilities and identify which CN will finish the job earliest. Therefore,

$$tx_j = min\ (\overline{tx_j}) \tag{5}$$

The completion time of jobs is always equal to the earliest possible start time plus the execution time, which is described as:

$$\overline{tc_j} = \overline{ts_j} + \overline{tx_j} \tag{6}$$

The earliest possible start time for Job Jj on a CN is the latest free time of the chosen CN if there are still jobs running on it. if there is no job running on the chosen CN ahead of job Jj's arrival. Jj can be executed on this CN immediately.

$$\overline{ts_j} = \{max\left(t_j, max\left(\overline{te_p}\right)\right), \forall p, \overline{P(p)} = \overline{P(j)}, p < j\} \tag{7}$$

3.2 Job Replication Approach

Menno Dobber et al. [11] analyze and compare the effectiveness of the dynamic load balancing and job replication approach. The two main techniques that are most suitable with the dynamic nature of the grid are Dynamic Load Balancing (DLB) and job replication (JR) [11]. Consider a grid system P with n computing node; a set J of jobs Ji (as given in equation 1, and 8) is considered to be run on P. As the name implies a JR scheme creates several replica of an individual job and schedule them to run into different nodes. The node that finish the job first, send a message to other node involve in a grid system to finish the execution of current job and start the execution of next job available in the queue.

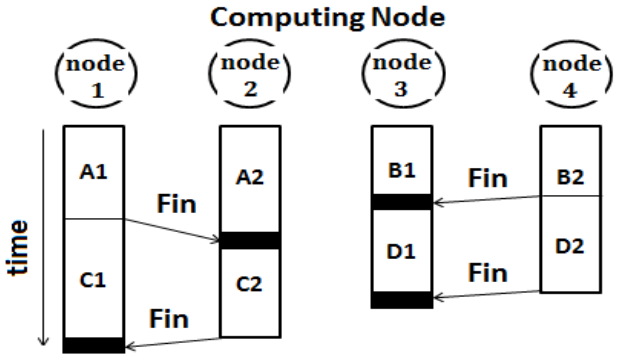

Figure 2. Job Replication (2JR) Scheme

A m-JR scheme creates m-1 precise copies of each job and these jobs are considered to be run on P. The same data set and the parameters are provided to the two copies of a job, to perform exactly the same computation so the calculations are completely the same. The JR approach consists of I iteration and one iteration takes total R-steps. N copies of all jobs have been spread out to P. As soon as the computing node finished one of the copies of the job, it sends a message to other computing nodes to kill

the current job execution and start the processing of the next job. A 2-JR scheme is show in fig. 2, consisting 4 CN and 4 jobs. Firstly 2 copies of each job each job is created after then each job with its copy are distributed to n =2 CN. In figure 3, one copy (A2) of job A1 is created and then distributed to CN1 and CN2. CN1 finished job A1 first, so it send a finalize message "fin" to CN2. Sending a message between CN take some network delay, therefore scheduling of next job by a CN can takes some time. Duration of a specific job is defined as:

$$\text{Job-Time} = \min\{\text{all its job time}\} + \text{possible send time} \quad (8)$$

3.3 Genetic algorithm Based Approach

Genetic algorithms (GA) [4, 7, 8, 22] are increasingly popular for solving optimization, search and machine learning problems. It is basically a well-known and robust search heuristics. It search optimal solution from entire search space In grid environment scheduling is a type of be NP complete problem, i.e. there is no known polynomial time algorithm to optimally solve this problem. The main objective to use GA for load balancing is to achieve the minimum of makespan (the latest completion time among all the jobs processed in CN), maximum node utilization and a well-balanced load across all the CNs. Genetic algorithms are well suited to solving scheduling problems, because unlike heuristic methods GA operate on a population of solutions rather than a single solution. A combination of intelligent agents and multi-agent approaches is applied to both local grid resource scheduling and global grid load balancing. GA is basically used to generate solutions related to optimization problems using various techniques such as selection, mutation, and crossover. Let P be a set, then the cost function CF, is given as:

$$CF : \mathrm{P} \rightarrow \mathrm{IR} \quad (9)$$

GA start with an initial set of random solutions called population. Each individual in a population is called a chromosome, represents a solution to the problem. The evaluation of chromosomes is done through generations. During each generation, the chromosomes are evaluated, using some fitness function. For creation of the next generation, new chromosomes, called offspring, are formed. The offspring is created by either using a crossover operator or using a mutation operator. The new generation is formed by selecting the individual using fitness values. After several generations, best chromosome is chooses, which represents the optimum or suboptimal solution to the problem [4]. The GA concentrates on an overall performance over a list of jobs and aims at a more desirable load balance across all the nodes in a computational grid.

3.4 Fuzzy Based Approach

This section introduces the fuzzy method used in the fuzzy load balancing algorithm. Fuzzy logic [9, 10, 12, 13, 23] deals with reasoning that is approximate rather than fixed and precise. It is a superset of conventional Boolean logic that has been extended to handle the concept of partial truth-values between "completely-truth", and "completely-false". In a more extensive sense, fuzzy logic is associated with the concept of fuzzy sets. Terms in the fuzzy set are given linguistic variables. Using fuzzy logic, one can specify the degrees of overload or otherwise with linguistic labels such as lightly loaded, normal, overloaded, heavily overloaded etc.. The fig 3(a) Show the use of fuzzy logic to represent degree of truth. The fuzzy expert system makes scheduling decisions based on a fuzzy logic. As shown in fig 3(b), if processor load is regarded as a linguistic variable, it may have the following terms as its values: light, moderate, and heavy. If 'light' is interpreted as a load below about one job, 'moderate' as a load between about 3 and 5 jobs and 'heavy' as a load above about 7 jobs, these terms may be described as fuzzy sets, where p represents the grade of membership [9]. A given fuzzy rule Ri consists of two differentiated parts, namely, antecedent and consequent parts, related to fuzzy concepts [12]. Rules activation conditions are reflected in the antecedent part of the rule. A rule within this is represented by the following expression:

$$R_i = \text{if } \omega_1 \text{ is } A_{1n} \text{ and / or} \ldots \omega_m \text{ is } A_{mn} \text{ then y is } B_n \qquad (14)$$

Where ω_m represents a system feature, y depicts the output variable and A_{mn} and B_n correspond to the fuzzy sets associated to feature m and output, respectively.

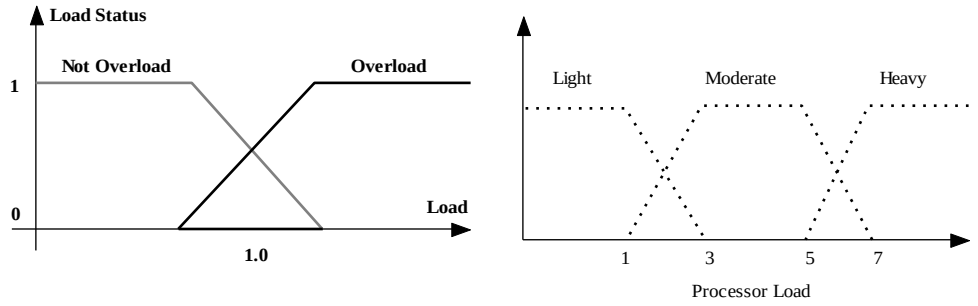

Figure 3. (a) Degree of truth representation (b) A linguistic variable, processor load

4. COMPARATIVE ANALYSIS

This section This section examine various characteristic of load balancing policies. Genetic algorithms are applicable to a wide variety of application [8]. It works better when we have to schedule a large number of jobs. The sliding windows technique [4,7] is generally used to trigger the GA. Sliding window consist a series of job for node assignment to schedule. The main

focused is about the size of sliding window and when these jobs have been assigned to appropriate node, the update of window is take place.

Table 1. Comparative analysis of load balancing policies

Approach	Factors to consideration	Advantages
Genetic Algorithm	1. Windows size and their update. 2. String selection for search node in search space. 3. Fitness function for measure the performance of string. 4. Population Size. 5. Processing overhead.	1. Minimum execution time. 2. Minimize Communication cost. 3. Maximum utilization of node. 4. Maximum throughput value. 5. Minimize makespan value.
Fuzzy Logic	1. Interference Engine. 2. Decision Making 3. Load Assignment. 4. State Update Decision.	1. Better Performance and throughput. 2. Response time is significantly better.
Job Replication	1. No. of copies of Job. 2. Communication between Nodes. 3. Processing overhead.	1. Consistently perform better when measure statistic is less than threshold value.
FCFS	1. Instantaneous decision. 2. Sequence of Job arrival.	1. Reduce the system response time. 2. Shorter Makespan.

Table 1 shows the comparative analysis of some load balancing policies. The measurement of GA is based on the quality of solution it produced after several generations. The measurements of input variables of a fuzzy controller must be properly combined with relevant fuzzy information rules to make inferences regarding the system Load State. The job replication scheme consist multiple copies of each job so it gives extra overhead to the computing node. To make an instantaneous decision, the FCFS approach is preferred. The primary focus of all the load balancing approaches is spread the workload to each node in such a manner that, the makespan is minimized and a well-balanced load among all the nodes in grid system therefore the current workload in the system must be considered to schedule the job to appropriate node.

5. CHALLENGES AND KEY ISSUE

5.1 Challenges in Load Balancing

There are various strategies have been developed for solving load balancing problem but it is not yet solved completely. Data partitioning and load balancing are weighty components of parallel computations. For static and dynamic load balancing various strategies have been developed, such as recursive bisection (RB) methods, space-filling curve (SFC) partitioning and graph partitioning [20]. A computation grid system consists of various components such as computing node and workstations, etc. There exist a heterogeneity between the various factors of a node such hardware architecture, physical memory, CPU speed and node capacity which affect the processing result. The dynamic behavior, and node failure can decrease the performance of grid, while the selection of resources (or node) for jobs could also be a factor.

5.2 Key Issues in Load Balancing

Load balancing is very crucial issue for the efficient operation of computational grid environments for sharing of heterogeneous resources, hence affect quality of service and scheduling. In dynamic load balancing, load sharing and task migration are some of the widely researched issues [7]. In a networked environment, interoperability (common protocols) is the central issue to be addressed. How to select efficient nodes is one of the issues of further investigation. As Grid is a distributed system utilizing idle nodes scattered in every region, the most critical issue pertaining to distributed systems is how to integrate and apply every computer resource (node) into a distributed system, so as to achieve the goals of enhancing performance, resource sharing, extensibility, and increase availability [15]. In order to make optimal balancing and work distribution decisions in Grid environment, a load balancer needs to take some or all of the following information into consideration:

- The capacity on each node (CPU, memory, and disk space, etc.)
- Current workload of each node
- Required capacity for each task
- Network connectivity and delay
- The assignment of processes to processors
- The use of multiprogramming on individual processors
- The actual dispatching of a process.

The choice of parameter and state information should also be considered. For various software engineering related issues, there are some software toolkit exists to provide effective solution. The distribution of load among the CN in an optimal way is not easy task, as it requires complete analysis of available resources and job. A load balancing policy can either be static

or dynamic. The major concern in static load balancing policy is to determine the execution time of the job, communication delays, and the resources used by computing node. Since an accurate estimation is not possible earlier in time, so emphasis can be done on to estimation of such quantities closer to accurate values.

6. CONCLUSIONS

In this paper, we discussed few contemporary load balancing strategies based on various approaches for computational grid environment. With the rapid development of technology, grid computing have increasingly becomes an attractive computing platform for a variety of applications. In a computational grid environment, load balancing is the process of improving the performance of system through re-distribution of load among the computing nodes. When the newly created jobs arbitrary arrive into the system, the node can become heavily loaded while other become either ideal or lightly loaded, therefore the job assignment and load sharing must be done carefully. The problem of load balancing is closely related to scheduling and allocation of job to computing node. For efficient utilization of grid resource and maximum node utilization, special scheduling policy is needed. Load balancing methods will vary greatly between different grid environment depending on the needs and the availability of computing node to perform the task. There are various factors which affects the performance of grid application such as load balancing, resource sharing, and resource heterogeneity therefore it must be considered for making decision.

REFERENCES

[1] I. Foster, C. Kesselman, and S. Tuecke, *The anatomy of the grid: Enabling scalable virtual organizations*, The International Journal of High Performance Computing Applications, 15 (3) (2001) 200-222.

[2] Rajkumar Buyya, and Srikumar Venugopal, *A Gentle Introduction to Grid Computing and Technologies*, Computer Socity of India, CSI Communication, July (2005).

[3] C. Gary Rommel, *The Probability of Load Balancing Success in a Homogeneous Network*, IEEE transactions on Software Engineering, 17(9), Sept. (1991) 922-933.

[4] Yajun Li, Yuhang Yang, Maode Ma, and Liang Zhoy, *A hybrid load balancing strategy of sequential tasks for grid computing environments*, Future Generation Computer Systems, 25 (2009) 819-828.

[5] Rajkumar Buyya and Manzur Murshed, *GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing*, The Journal of Concurrency and Computation: Practice and Experience (CCPE), Volume 14, Issue 13-15, Wiley Press, Nov.-Dec., (2002).

[6] Vandy Berten, Joel Goossens, and Emmanuel Jeannot, *On the Distribution of Sequential Jobs in Random Brokering for Heterogeneous Computational Grids*, IEEE Transections on Parallel and Distributed System 17 (2) (2006) 1-12.

[7] Albert Y. Zomaya, Yee-Hwei Teh, *Observations on Using Genetic Algorithms for Dynamic Load-Balancing*, IEEE Transections on Parallel and Distributed System, 12(9), Sept. (2001) 899-911.

[8] Carlos Alberto Gonzalez Pico, Roger L. Wainwright, Dynamic Scheduling of Computer Tasks Using Genetic Algorithms, *Proc. of the First IEEE Conference on Evolutionary Computation - IEEE World Congress on Computational Intelligence*, Orlando, Florida June, (1994) 829-833.

[9] Chulhye Park, Jon G. Kuhl, *A Fuzzy-Based Distributed Load Balancing Algorithm for Large Distributed Systems*, IEEE (1995) 266-273.

[10] Kaveh Abani, Kiumi Akingbehh, Adnan Shaout, *Fuzzy Decision Making for Load Balancing in a Distributed System*, IEEE, (1993) 500-502.

[11] Menno Dobber, Rob van der Mei, Ger Koole, *Dynamic Load Balancing and Job Replication in a Global-Scale Grid Environment: A Comparison*, IEEE Transections on Parallel and Distributed System, 20(2), Feb. (2009) 207-218.

[12] Yu-Kwong Kwok, Lap-Sun Cheung, *A new fuzzy-decision based load balancing system for distributed object computing*, Journal of Parallel and Distributed Computing, 64 (2004) 238–253.

[13] Mika Rantonen, Tapio Frantti, Kauko Leiviska, *Fuzzy expert system for load balancing in symmetric multiprocessor systems*, Expert Systems with Applications, 37 (2010) 8711–8720.

[14] Junwei Caoa, Daniel P. Spooner, Stephen A. Jarvis and Graham R. Nudd, *Grid load balancing using intelligent agents*, Future Generation Computer Systems, 21 (2005).

[15] K.Q. Yan, S.C. Wang, C.P. Chang, J.S. Lin, *A hybrid load balancing policy underlying grid computing environment*, Computer Standards & Interfaces 29 (2007) 161–173.

[16] Jun Wang, Jian-Wen Chen, Yong-Liang Wang, Di Zheng, *Intelligent Load Balancing Strategies for Complex Distributed Simulation Applications*, 2009 International Conference on Computational Intelligence and Security, 2009 (182-186).

[17] Kuo-Qin Yan, Shun-Sheng Wang, Shu-Ching Wang, Chiu-Ping Chang, *Towards a hybrid load balancing policy in grid computing system*, Expert Systems with Applications 36 (2009), 12054–12064.

[18] Brighten Godfrey, Karthik Lakshminarayanan, Sonesh Surana, Richard Karp, Ion Stoica, *Load Balancing in Dynamic Structured P2P Systems*, IEEE INFOCOM (2004).

[19] Luis Miguel Campos, Isaac D. Scherson, *Rate of change load balancing in distributed and parallel systems*, Parallel Computing 26 (2000), 1213-1230.

[20] Karen D. Devine, Erik G.Boman, Robert T. Heaphy, Bruce A. Hendrickson, James D. Teresco, Jamal Faik, Joseph E. Flaherty, Luis G. Gervasio, *New challenges in dynamic load balancing*, Applied Numerical Mathematics 52 (2005) 133–152.

[21] Satish Penmatsa, and Anthony T. Chronopoulos, *Comparison of Price-based Static and Dynamic Job Allocation Schemes for Grid Computing Systems*, Eighth IEEE International Symposium on Network Computing and Applications, (2009) 66-73.

[22] In Lee, Riyaz Sikora, and Michael J. Shaw, *A Genetic Algorithm-Based Approach to Flexible Flow-Line Scheduling with Variable Lot Sizes*, IEEE Transactions On Systems, Man, a Cybernetics-Part B: Cybernetics, 27(1), Feb. (1997).

[23] S.Salleh, and A.Y.Zomaya, Using Fuzzy Logic for Task Scheduling in Multiprocessor Systems, *Proc. 8th ISCA International Conference on Parallel and Distributed Computing Systems*, Orlando, Florida, (1995) 45-51.