An Optimal Job Selection method in Load Balancing Algorithms of Economical Grids

Mohsen Amini Salehi[1], Hamid Tabatabaee Yazdi[2], Mohammad Reza Akbarzade Toutoonchi[3]

[1] Department of Software Engineering, Faculty of Engineering, Islamic Azad University, Mashhad Branch, Iran.

[2] Department of Software Engineering, Islamic Azad University, Ghouchan Branch, Iran.

[3] Department of Electric and Electronic Engineering Faculty, Ferdowsi University, Mashhad, Iran

Amini@mshdiau.ac.ir, ht@ferdowsi.um.ac.ir, Akbarza@ieee.org

Abstract

A computational grid is a widespread computing environment that provides huge computational power for large-scale distributed applications. Load balancing, has a considerable effect on the grid middleware performance. Load-balancing algorithm and job selection are two parts of any load balancing methods in the grid. In the previous work we had proposed a Load-balancing algorithm for the grid environment. In this paper, we intend to complete the load-balancing algorithm by proposing an optimal job selection algorithm. As selecting jobs is a multi-criteria problem in nature, we define the job selection algorithm as an optimization problem and solve it using GA. The performance and the optimality of the method is proved by relevant simulations and experimental results.

Key Words:
Grid computing, Load balancing, Job Selection, Genetic Algorithms

1. Introduction

Grid computing enables users to gather geographically distributed computational resources to create a huge virtual supercomputer that can be applied for executing computational-intensive programs [1, 2]. An ideal grid environment should provide access to all the available resources seamlessly and fairly.

A resource manager is an important infrastructural component of a grid computing environment. The overall aim of resource management is to efficiently schedule applications that need to utilize the available resources in the grid environment [3, 4]. An ideal resource manager should exploit all of the grid resources uniformly [5, 6]. Considering the largeness, dynamic resources, stochastic users and other specifications of the grid, it is impossible to use resources in equilibrium, unless using load balancing/ sharing methods [7].

Load balancing algorithms try to equalize the workload among all available nodes, while load sharing algorithms just try to assure that no node is at rest whereas elsewhere processes are waiting for service [7].

Generally, each load balancing and load sharing algorithm can be defined by three rules: the location rule, the distribution rule and the selection rule [8]. The location rule determines which domain of resources will be included in the balancing operation. The domain may be local, i.e. inside the virtual organizations (VO), or global, i.e. between different VO. The distribution rule decides how to redistribute the workload among available resources on the domain. The selection rule decides on the picking proper jobs from overloaded node for reassign to the underloaded. So far, many contributions have been done to provide a good load balancing/sharing method for the Grid. A good survey on these algorithms can be found in [9].

In our previous works [3, 4], we have proposed new mechanisms for load balancing in the Grid. However, like most of other contributions, our mechanism had not any intelligence in picking proper jobs for transfer. In other words, they only propose how to find overloaded and underloaded nodes in the grid environment.

In this paper, we aim to provide a new job selection algorithm in which an optimal set of jobs are selected for transmission to underloaded node.

The rest of the paper is organized as follows: Section 2 contains a literature survey on job selection methods. In Section 3, the new job selection algorithm is described. The performance metrics and simulation results are included in Section 4. At the end, we present the conclusion of the article as well as the future works which can be done in the same direction as the inclination of this research.

2. Problem Definition

As mentioned above, Load balancing methods are designed essentially to spread the load on resources equally and maximize their utilization while minimizing the total task execution time [8]. Selecting the optimal set of jobs for transferring has a significant role on the efficiency of the load balancing method as well as grid resource utilization. This problem has been neglected by researchers in most of previous contributions on load balancing, either in distributed systems or in the grid.

In preceding mechanisms, jobs, in the overloaded node, were simply selected from the end of ready queue until it fills the capacity of underloaded nodes [12]. However, such selection mechanism, in one hand, may consequences to a high communication overhead, lengthening the execution time and decreasing efficiency while, on the other hand, in economical grids, where resources receive money on executing other jobs, it may even result in losing money.

Such reasons have made us to propose a job selection method in which jobs are selected not only based on their position in the ready queue, but other factors like communication overhead and proficiency are considered.

Genetic Algorithm (GA) is a search algorithm rooted in the principles of natural genetics. GAs join the exploitation of past results with the exploration of new areas of the search space. By using *survival of the fittest* techniques, a GA can imitate some features of a human search. A generation is a group of artificial creatures (strings). In every new generation, a set of strings is produced using information from the prior ones. GAs are randomized, but they are not simple random walks. They use historical information to guess on new search points with expected enhancement [8, 10, 11].

In the next section, the selection algorithm is posed as an optimization problem and genetic algorithm is employed as a heuristics to combat the problem.

3. Algorithm Outline

As stated earlier, in this paper, we intend to put forward an algorithm which determines the optimal subset of jobs for transfer in a load balancing method.

At the first step, criteria based on which optimality is defined should be determined.

Considering difficulties of the grid environments, like communication overhead and dynamic bandwidth of routs, it seems that smaller jobs are better choice for transmission. In other words, larger jobs take longer time for transmission. Even, sometimes it is better to execute job locally instead of send it to another node and execute there. Therefore, job size is a decisive criterion for picking or not picking a job.

Since financial issues are important in the practical grid and different VOs compete with each other based on economical profitability [13], hence, in load balancing condition, the sender node would prefer to select less profit jobs for transmission.

Waiting time is the other decisive factor for selecting jobs to be relocated. Obviously, preventing starvation is a crucial goal of each load balancing/sharing algorithm. Therefore, the sender would rather to displace jobs which have more waiting time.

Now, we are going to select jobs from the overloaded nodes based on the combination of the aforementioned optimality criteria. It goes without saying that, selecting jobs from the end of ready queue wouldn't satisfy these criteria at all. On the other hand, if there are N jobs in the ready queue, then, we should examine 2^N different conditions and select the best (optimal) set of jobs that satisfies all of above criteria. Unarguably, this brute force algorithm is NP-Complete and the time complexity is not acceptable. Hence, it is better to use heuristic methods to find the optimal or near optimal solution in a satisfactory time.

Since the problem is a combinatorial optimization problem, GA is a good heuristic that help us combat the problem [12]. The technique needs a coding scheme that can represent all legal solutions to the optimization problem. Any possible solution is uniquely represented by a particular string which is called chromosome [14]. Here, the chromosomes are in decimal form and each gene shows a job number. Figure.1 shows a sample chromosome. These chromosomes are manipulated by crossover and mutation operators until the algorithm converges on an optimal or near optimal solution.

10	0	0	0	7	41	0	44	38	3

Figure.1. A sample chromosome. Each entry of the border indicates a gene and shows a job number. Job numbers are unique in each chromosome except zeros. Zeros show that there isn't any job in that cell.

A method that assigns a quality value to each solution string (*fitness function*) guarantee that this operation proceed in correct order. We formalize the proposed criteria in the form of the following fitness function for each chromosome C_k:

$$f_{C_k} = \sum_{i=1}^{L} (W_i * \frac{1}{S_i} * \frac{1}{P_i}) - \left(\sum_{i=1}^{L} T_i - \alpha \right)^2 \quad (1)$$

Where *'L'* is the chromosome length, W_i indicated the waiting time of each job *i* (gene *i*) in the ready queue, S_i shows the size of each job, P_i is the profit attain from execution of each job, T_i is the time needed for executing a job, and α is the amount of time the receiver can give to the coming jobs. Actually, α is a constraint from the receiver that should be satisfied by the sender. This constraint is fulfilled by the penalty function:

$$\left(\sum_{i=1}^{L} T_i - \alpha\right)^2 \quad (2)$$

GA starts from an initial population and uses simple cross over and mutation operations in each generation it to maximize the fitness function.

4. Experimental Results

In this section, the simulation environment is described then we investigate several common statistics to illustrate the performance of the mechanism described.

We have used Genetic algorithm library of Matlab for simulation. In this simulation we assume the maximum number of jobs waiting in the ready queue is 100 and the maximum number of jobs could be selected are 50. A structure is used to indicate each job. α, which is the capacity of the receiver, could be differs time to time.

We have also implemented two other methods. The first one is the method which selects jobs from the end of ready queue. We briefly call this method *'MaxWait'*. The second method, which is implemented, builds all possible subsets and, finally, selects the best one. We call this method *'SubSets'*.

In the first experiment, fitness attained from each method is reported. As illustrated in Figure.2, the *SubSets* method results in the best fitness while the *MaxWait* method has the poorest fitness. The fitness reached through our *GA* method, after 50 and 100 generation, is near to the optimal one (*Subsets* method)

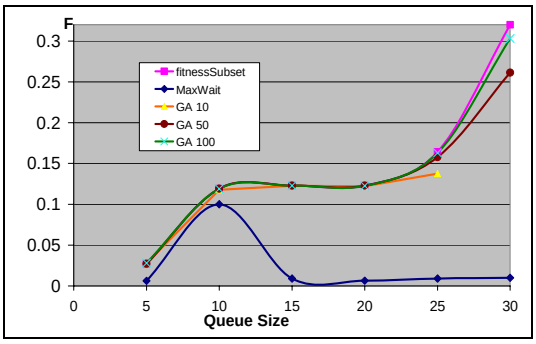

Figure.2. Comparing the fitness of selected jobs (F) for different Queue Sizes in each of three methods.

Time consumption of proposed *GA* method, with 50 and 100 generation, versus *Subsets* is compared in the second experiment. The NP-completeness of *Subsets* method could be easily understood from Figure.3. However, as it can be seen in this figure, the new method eliminates this unacceptable time. As figure 3 could not clarify the time difference between various methods, results of this experiment are also presented in Table 1.

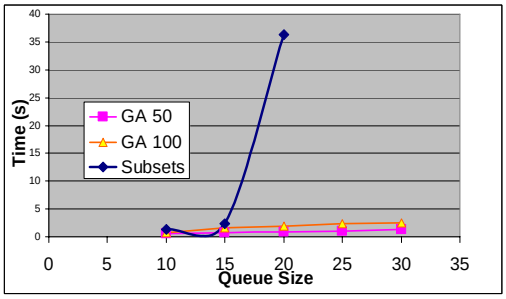

Figure 3. Time needed for selecting jobs for our new method (*GA*) (base on seconds) versus the optimal algorithm (*Subsets*)

Queue Size	Subsets	GA (50 Generation)	GA (100 Generation)
10	1.401	0.591	0.800
15	2.402	0.665	1.681
20	36.252	0.812	1.854
25	519.547	0.988	2.298
30	8304.21	1.310	2.532

Table 1. Precise time consumed (base on seconds) for selecting jobs in queues with different sizes in each method.

The third experiment elucidates the Efficiency of each method. Efficiency is computed based on its standard rule i.e. Performance/ Cost. Here, performance is the fitness achieved and cost is the time needed to reach that fitness. Since *MaxWait* method behaves completely random, we do not count on it in this experiment. However, the efficiency of other methods is illuminated in Figure 4.

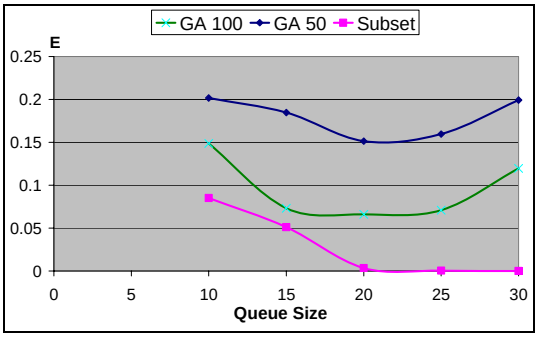

Figure 4. Efficiency (*E*) achieved in each method for queues with different sizes.

As grids are going to be economical [13], it is very important for any load balancing algorithm to get rid of less benefit jobs. In our new method, we have considered this factor and try to lessen the profit of submitted jobs. Hence, as our last experiment, profit loss of each method is revealed in figure 5. To achieve this experiment, a random number is assigned to each job as its profit.

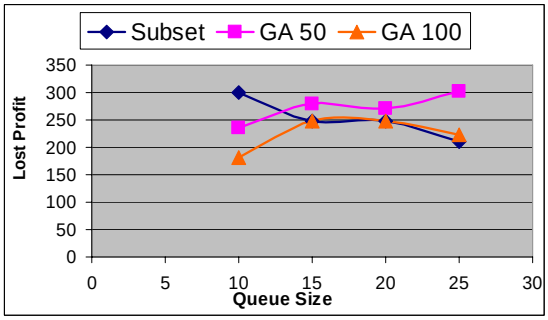

Figure 5. Lost profit caused by job selection algorithm for different queue sizes.

5. Conclusion and Future Works

As described in this paper, selecting an optimal set of jobs for transmission in overloaded nodes is critical for load balancing in the grid. In this way, with respect to grid specifications, an algorithm which acts based on genetic algorithms is proposed in this paper to meet the challenges of picking proper jobs for transmission. Experimental results, which are shown in preceding section, prove the appropriateness of genetic algorithm for solving this problem.

In future work, we plan to implement this method in a more realistic environment. Promoting intelligence and more adaptation, establishing billing contracts among resources as they exchange their extra loads, as well as overcoming security concerns are other future work.

References:

[1] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal of Supercomputer Applications*, 2001.

[2] Anand Padmanabhan, Shaowen Wang Sukumar Ghosh, and Ransom Briggs, "A Self-Organized Grouping (SOG) Method for Efficient Grid Resource Discovery", IEEE Grid Computing Workshop 2005, pp. 312-317.

[3] Mohsen Amini Salehi, Hossain Deldari, Bahare Mokarram Dorri, "MLBM: Multi-Level Load Balancing Mechanism in Agent-based Grid," 8th International Conference on Distributed Computing and Networking (ICDCN06), Lecture Notes in Computer Science, Dec 2006, India.pp.47-53.

[4] Mohsen Amini Salehi, Hossain Deldari, "ADAPTIVE FUZZY ECHO SYSTEM OF ANTS IN GRID LOAD BALANCING," Advances in Computational Sciences and Technology Journal (ACST) (ISSN 0973-6107) (Accepted on 28/AUG/2007 but it is not published yet).

[5] J. Cao, Self-Organizing Agents for Grid Load Balancing, in "Proc. 5th IEEE/ACM Int. Workshop on Grid Computing (GRID'04).

[6] G. Nudd, D. Kerbyson, E. Papaefstathiou, S.C. Perry, J.S. Harper, and D. V. Wilcox, PACE a toolset for the performance prediction of parallel and distributed systems, Int. J. High Performance Computing Applications 3(2000), pp. 228-251.

[7] Orly remien, Jeff Kramer. Methodical analysis of adaptive load sharing algorithms, IEEE Transactions on Parallel and Distributed Systems 3(Nov.1992), pp.747-760.

[8] A. Y. Zomaya and Y. The, Observations on using genetic algorithms for dynamic load-balancing, IEEE Transactions on Parallel and Distributed Systems 9 (September 2001), pp. 899-911.

[9] Montresor, et al, "Messor: Load-Balancing through a Swarm of Autonomous Agents", Proc. of 1st Int. Workshop on Agents and Peer-to-Peer Computing, Italy, 2002.pp.

[10] D.E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", Reading, Mass.: Addison-Wesley, 1989.

[11] M.D. Kidwell and D.J. Cook, "Genetic Algorithm for Dynamic Task Scheduling," Proc. IEEE 13th Ann. Int'l Phoenix Conf. Computers and Comm., pp. 61-67, 1994.

[12] Krzysztof Kurowski1, Jarek Nabrzyski1, Ariel Oleksiak1, Jan Węglarz1, "Scheduling Jobs on the Grid – Multicriteria Approach," Journal of COMPUTATIONAL METHODS IN SCIENCE AND TECHNOLOGY Vol 12, No. 2, pp. 123-138, 2006

[13] K. Krauter, R. Buyya, and M. Maheswaran, A Taxonomy and Survey of Grid Resource Management Systems, Technical Report, Monash University, Australia, 2000.

[14] J. Cao, D. P. Spooner, S. A. Jarvis, S. Saini, and G. R. Nudd, "Agent-Based Grid Load Balancing Using Performance-Driven Task Scheduling," in Proc. of 17th IEEE Int. Parallel and Distributed Processing Symp. Nice, pp. 218-224, 2003.