International Journal of Innovative Research in Science and Engineering
Vol. No.2, Issue 02, February 2016
www.ijirse.com
ISSN: 2454-9665

RELIABLE JOB SCHEDULING FOR GRID ENVIRONMENT

Mohanraj.P[1], Alagar.A[2], Arun Kumar K.G[3],

[1,2,3]*Dept. of Computer Science and Engineering , Excel Engineering College (India)*

ABSTRACT

Grid computing is a high performance computing environment to solve larger scale computational demands. It contains resource management, job scheduling, and information management. Job scheduling is a fundamental issue in achieving high performance in grid computing systems. Hence scheduling the jobs to resource in grid computing is complicated due to the distributed and heterogeneous nature of resources. The available techniques are based on scheduling policies and few researches are working towards heuristic approach .This project proposes a genetic optimization algorithm for job scheduling. This strategy maintains the fault occurrence history of resources in Grid Information Server (GIS). Whenever a resource broker has job to schedule it uses the resource fault occurrence history information from GIS and depending on this information fitness function is calculated. Genetic Algorithm with RFOH finds a near optimal solution for the problem. This proposed algorithm will decrease the probability of failure and increases the reliability for high degree of user satisfaction.

I. INTRODUCTION

Computer scientists in the mid-1990s began exploring the design and development of an analogous infrastructure called the computational power Grid [1]. A Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed autonomous and heterogeneous resources dynamically at runtime, depending on their availability, capability, performance, cost, and users' quality-of-service requirements [3]. Resources can be computers, memories, instruments (such as telescope), software applications, and data that all are connected through the Internet.

In Grid environments, there are many jobs to schedule for parallel execution on this system. Since that scheduling problem is a NP-Hard problem, we can use evolutionary algorithms to solve it. Among these algorithms, genetic Algorithm (GA) is more usual which is used in this paper. GA is a global search technique, which maintains a pool of potential solutions called Chromosome [4]. The GA produces new solutions through combining the good features of existing solutions randomly. In this algorithm, there is an operation named Crossover operation for global searches through the solution space by randomly exchanging portions of two chromosomes. Another important local search operator is Mutation, which works by randomly changing one of the genes in a chromosome. The whole process is repeated number of times, called generations or iterations.

Typically, the probability of a failure is higher in the grid computing than in a traditional parallel computing [5, 6] and the failure of resources affects job execution fatally. Therefore, a fault tolerance service is essential in computational grids [5]. One of important goals in distributed systems such as Grid is to construct the system in such a way that it can automatically recover from error without serious decreasing of system performance.

For having reliable and thereupon fault tolerant job scheduling in the Grid, we propose a new strategy. In this strategy, Resource Fault Occurrence History (RFOH) information is used in GA. Hence this causes achieving a suitable solution for scheduling problem, which has reliability too. The reason is that we try to reduce the selection probability of resources with more fault occurrence history.

Rest of paper is organized as following: Section 2 contains the description of the related work. In section 3, proposed strategy is described. Section 4 discusses the simulation results and finally Section 5 concludes the paper.

II. RELATEDWORK

The probability of a failure in large-scale Grids is much greater than traditional parallel systems [5, 6]. Thus, Grid system should be able to identify and manage faults and support reliable execution of jobs. Grid System failure handling techniques are classified as task-level and workflow-level [7]. Task-level techniques mask the effects of the execution failure of tasks in the Grid system, while workflow-level techniques manipulate the system structure such as execution flow to deal with erroneous conditions. Checkpoint technique is one of the task-level techniques. This technique moves failed tasks transparently to other resources, so that the task can continue its execution from the point of failure [8].

In [8] with using GA, a solution was proposed for allocating jobs to resources by Grid scheduler. Also for having fault tolerant job scheduling, Checkpoint technique was used. But RFOH information wasn’t considered as what we used in our strategy.

In [6] the history of the fault occurrence of resource is maintained in GIS. Whenever a resource broker has a job to schedule it uses this information from GIS and depending on this information, it uses different intensity of Check pointing considering that resources have different tendency towards fault. In that way, only quality of Check pointing is discussed and there is no attention to the type of scheduling. But, we used this information in other format within GA for having reliable and thereupon fault tolerant job scheduler.

In this paper, we use RFOH information in GA. doing this; we can reduce the selection probability of the resources with more fault occurrence history. Therefore we have a reliable scheduling and kind of fault tolerance. Further we could have user satisfaction in job scheduling, too. Hence we have a combination of reliability and user satisfaction.

III. OUR NEW APPROACH

According to Fig. 1, our proposed strategy consists of three components: (a) Fault Detector, (b) RFOH Storage, and (c) Job Scheduler. Each of these components is explained at following.

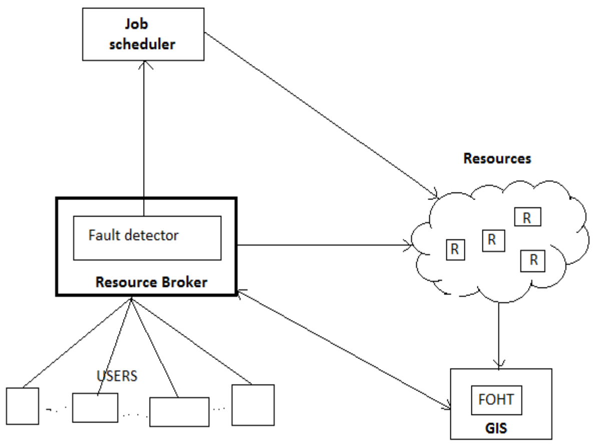


Figure 1 Architecture of our system

3.1 Fault Detector

A. **Fault Detector** Fault detection on the resources is done by the resource broker. After job allocation to the resource, the resource broker should receive a response of job execution from it within a certain time interval. If in this time interval, the resource broker could not get any response, it realizes that a fault occurred on that resource. Then in next step it sends the information about that fault to GIS. Mentioned time interval is a function of resource speed, communication latency, and queue length of the resource.

3.2 RFOH storage

For having a reliable job scheduler, we use RFOH in GA. This causes to reduce the selection probability of resources with more fault occurrence history. To store RFOH, GIS maintains the history of the fault occurred in resources in a table called Fault Occurrence History Table (FOHT) [9]. FOHT has two columns. First column presents the history of fault occurred in the resources and second column keeps the number of job execution by resources. So the number of rows and the number of resources are equal. According to [9], FOHT is updated when:

1) The resource is unable to execute the given job in the specified deadline. Then fault index of this resource (first cell) is incremented by 1.

2) A job is allocated to the resource. Then number of job execution by this resource (second cell) is incremented by 1.

Therefore FOHT is updated with the following equation for ith resource [9].

$$\text{FOHT}[i] = \begin{cases} \text{FOHT}[i, 1] + 1\text{: with fault occurrence in } i^{th} \text{ resource} \\ \text{FOHT}[i, 1] + 1\text{: with job allocation to } i^{th} \text{ resource} \end{cases}$$

Fig. 2 presents a part of FOHT in a given time. For example, this figure indicates the number of job execution by R1 is 6700 but at ten times of these executions, fault occurred.

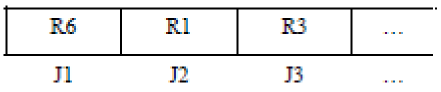
1	2

R1	10	6700
R2	5	100
R3	0	123
R4	8	120
.	.	.
.	.	.
.	.	.
.	.	.

Figure 2 A part of FOHT at given time

C. **Job scheduler** We assumed that there are n jobs such J_1, J_2, …, J_n and we want to allocate them to m resources such R_1, R_2, …, R_m. Such scheduling problem is a NP-Hard problem. So we use Genetic Algorithm to solve it. As mentioned before, for having reliable and fault tolerant job scheduling, we use RFOH in this algorithm. According to Fig. 3, this method has 5 steps which work as mentioned below:

1) Population initialization: At first we generate an initial population of chromosomes randomly. With regards to Fig. 4, each chromosome represents a possible solution, which is a mapping sequence between jobs and resources. In this method the length of chromosomes and the number of jobs in Grid are equal. According to Fig. 4, The J1, J2, and J3 jobs are executed by R6, R1, R3 resources respectively.



R6	R1	R3	…
J1	J2	J3	…

Figure 4. Representation of a part of a chromosome

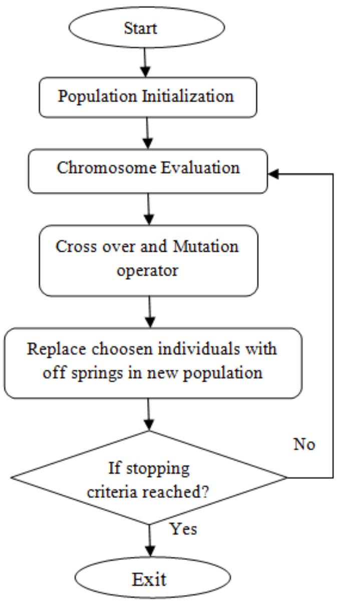

Start
Population Initialization
Chromosome Evaluation
Cross over and Mutation operator
Replace choosen individuals with off springs in new population
If stopping criteria reached?
No
Yes
Exit

Figure 3. The steps of our proposed strategy

2) Chromosome evaluation: Each chromosome must evaluate to specify its fitness value. This is implemented with fitness function which is defined based on below parameters.

a) Users Satisfaction: We assume that the considered parameter of user is the response time of job execution. So the fitness function is defined as below:

$$f_2 = \sum_{i=1}^{n} R_{ti}$$

In (2) iis Response Time of execution of R th job at one chromosome. According to this equation a chromosome with proper resources for job executing - that have less execution time – will have less fitness value in the population. So in this population whatever the fitness value of a chromosome becomes lesser than others, it will be a best solution for our problem.

b) Reliability: As we mentioned, for having a reliable job scheduler, we use RFOH in our strategy. By using RFOH information, the resources with more tendencies to failure will have less probability to select. So we can define the fitness function as below:

$$f_2 = \sum_{i=1}^{n}\sum_{i=1}^{n} \frac{FOHT[i,1]}{FOHT[i,2]} \times 100$$

According to (3), we must calculate the sum of percentage of RFOH to total number of job execution proportion for each selective resource with using FOHT. Hence, a chromosome with great fault occurrence probability will have great fitness value in the population. Therefore we attempt to minimize fitness value of chromosomes. This causes the reduction of selection probability of resources which are more tendencies to failure.

Now by combining (2) and (3), we can introduce a fitness function for chromosome evaluating as below:

$$f = \sum_{i=1}^{n} R_{ti} + \sum_{i=1}^{n} \frac{FOHT[i,1]}{FOHT[i,2]} \times 100$$

This equation shows that if the resources within a chromosome have less total response time for executing each job and less fault occurrence history, that chromosome will have a less fitness value. Considering that we attempt to minimize fitness value in the population, such chromosome will have great probability for selection as a solution.

3) Implement Crossover and Mutation operators: After evaluating chromosome for finding proper solution, we use Crossover and Mutation operators. There are various types of Crossover and Mutation operators. In this strategy, we used Two Points Crossover and a kind of Uniform Mutation. Two points Crossover operator selects a random pair of chromosomes and exchanges a random part of those. Our Mutation operator randomly selects a chromosome, and then randomly selects a job within the chromosome, and - according to first column of FOHT - if its resource has more than one fault occurrence history, reassigns it to a new resource randomly.

4) Replacement: After Crossover and Mutation performed, we replace offsprings with their parent chromosomes in the population.

5) So far, one iteration of algorithm is done. This algorithm stops when a predefined number of evolutions are reached, or all chromosomes converge to the same mapping, or no improvement in recent evaluations, or a cost bound is met [9].

Finally after stopping the algorithm, the chromosome with less fitness value is selected as a problem solution.

IV. SIMULATION RESULTS

In this section, we evaluate our proposed strategy in two steps. In both steps, we have used Matlab toolbox for Genetic Algorithms as a simulator that named “gatool”. The simulation parameters and settings are listed in Table1.

Number of jobs	120
Job Types (40 jobs in each group)	Short:1-10 instructions Middle:45-55 instructions Long:90-100 instructions
Number of resources	120
Resource types (24 node in each group)	Veryslow, Slow, Middle, Fast,Very fast
Resource failure rates	Very faulty:90%-100% fault occurence Faulty:45%-55% fault occurence Safe:0%-10% fault occurence

Table1. Simulation parameters and settings

Having 120 jobs for executing, with regards to Fig. 4, the general schema of chromosomes is as Fig. 5. According to step 1 in section B of part III, first a random population is generated. Then in step 2, each chromosome is evaluated according to (4). We can create FOHT and also calculate total execution time of each chromosome with attention to Table 1 settings. After that, in step 3 Crossover and Mutation operators are used in population and offsprings is replaced with their parents in step 4 and then step 2 is resumed. After the number of iterations, one of introduced conditions in step 5 will occur and cause to stop algorithm. In this time, a chromosome with lesser fitness value can be a near best solution for our problem.

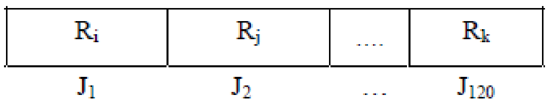


Figure 5. General schema of chromosomes in our example

D. Evaluation of total execution time

Simulation results illustrate that our proposed strategy can select proper resources for job executing with less execution time. For performance evaluation, we compared our new algorithm with the algorithm which never uses RFOH information. Fig. 6 represents that if during execution the fault doesn’t occur, the total execution time with our new strategy is less than total execution time without RFOH information. So the performance of GA with RFOH is better than simple GA in the case of total execution time.

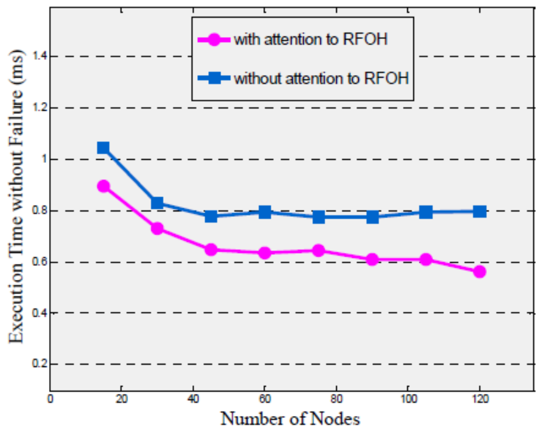


Figure 6. Total execution time comparison by varying number of resources

E. Evaluation of fault tolerance and reliability

According to fig. 7, our new strategy causes to reduce the probability percentage of fault occurrence in the selective resources for executing jobs. This figure shows that without RFOH information the probability of faulty resources selection is great. Furthermore, this probability reduces with decreasing the number of resources. But with our strategy it decreases.

Also fig. 8 represents that in simple GA without attention to RFOH information, the reliability decreases. But usage our new strategy increase the reliability. Therefore we have a rather reliable selection because the resources with more fault occurrence history weren't selected.

We must notice that despite if a resource has a kind of reliability, but it is a slow machine it is not suitable. So it isn't selected to execute one of our jobs. This shows that we can have a combination of user satisfaction and reliability.

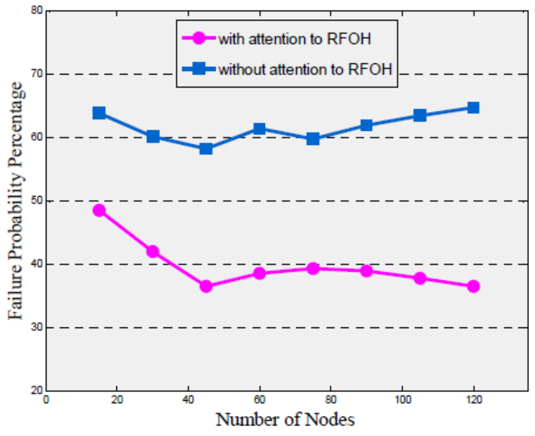


Figure 7. Failure probability percentage comparison by varying number of resources

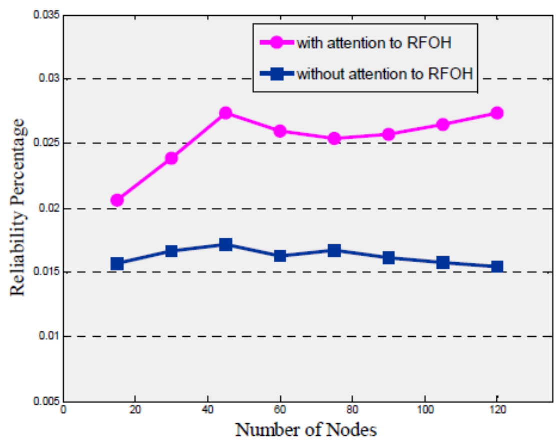


Figure 8. Reliability percentage comparison by varying number of resources

V. CONCLUSION

In Grid environments, task execution failures can occur for various reasons. In this paper we presented a new GA for reliable job scheduling in the Grid. This algorithm uses RFOH information which is maintained in FOHT. The using of this information causes the reduction of selecting chance of the resources which have more failure probability. Simulation results indicate that our proposed strategy decreases total time of job executing.

REFERENCES

[1] I. Foster, C. Kesselman, and S. Tueke, “The anatomy of the grid: Enabling scalable virtual organizations,” Supercomputing Applications, 2001.

[2] Foster and C. Kesselman, “The Grid Blueprint for a Future Computing Infrastructure,” San Mateo, CA: Morgan Kaufmann, 1999.

[3] M. Baker, R. Buyya and D. Laforenza , “Grids and Grid Technologies for Wide-area Distributed Computing,” Software-Practice & Experience, Vol. 32, No.15, 2002, pp: 1437-1466.

[4] A.Y. Zomaya, R.C. Lee, and S. Olariu , “An Introduction to Genetic-Based Scheduling in Parallel-Processor Systems,” Solutions to Parallel and Distributed Computing Problems: Lessons from Biological Science, A.Y. Zomaya, F. Ercal, and S. Olariu, eds., New York: Wiley, 2001, chapter 5, pp. 111-133.

[5] HwaMin Lee, KwangSik Chung, SungHo Chin, JongHyuk Lee, DaeWon Lee, 2005 ,"A resource management and fault tolerance services in grid computing", Journal of Parallel and Distributed Computing, Vol. 65, pp. 1305-1317.

[6] Babar Nazir, Taimoor Khan, “Fault Tolerant Job Scheduling in Computational Grid,” 2nd International Conference on Emerging Technologies Peshawar, Pakistan (IEEE—ICET), 2006 .

[7] S. Hwang and C. Kesselman “Grid Workflow: A Flexible Failure Handling Framework for the Grid,” In 12th IEEE International Symposium on High Performance Distributed Computing (HPDC’03), Seattle, Washington, USA, IEEE CS Press, Los Alamitos, CA, USA, June 22 - 24, 2003.

[8] S. Baghavathi Priya, M. Prakash, Dr. K. K. Dhwan, “Fault Tolerance-Genetic Algorithm for Grid Task Scheduling using Check Point,” The Sixth International Conference on Grid and Cooperative Computing (GCC), 2007.

[9] Leyli Mohammad Khanli, Maryam Etminan Far, and Amir Masoud Rahmani, “RFOH: a New Fault Tolerant Job Scheduler in Grid Computing”, The 2nd InternationalConference on Computer Engineering and Applications (ICCEA), Bali Island, Indonesia, March 19-21, 2010

[10]. R.A.M. Abd El-Hady “THE ABRASION RESISTANCE OF WARP-KNITTED FABRICS USED IN CAR SEAT COVERS” International Journal of Advance Research in Science and Engineering Vol. 5 No.1, 2319-8354.