A Reliable and Fault Tolerant Job Scheduling System in Market-Based Grids Using Case-Based Reasoning Method

Asgarali Bouyer

***Abstract*—In market-based Grid systems, a main aim is to execute jobs with considered quality of service requirements based on user defined budget. Since grid has heterogonous resource with unpredictable faults, the user cost constraints and expected service requirements may not provided. Therefore, using a better approach to resource scheduling to reduce fault is necessary. This paper presents a predictive approach on fault tolerance mechanisms for faultless job scheduling on market-based grids. The Case-Based Reasoning technique has been used for selecting fault tolerant nodes. This approach applies a specific structure in order to prepare fault tolerance between provider nodes to retain system in a safe state with minimum data transferring. Certainly, this algorithm increases fault tolerant confidence therefore, performance of grid will be high.**

***Index Terms*—Market-based grid, fault tolerance, case-based reasoning, job scheduling.**

I. Introduction

Grid computing is an amazing infrastructure to solve some problems that need to strong and heavy computation with very long time execution [1]. It is cooperation of different computers, for a specific task, so that the user acquires better performance for that specific task. In this environment, the resources are geographically distributed, but in logical aspect, these are as virtual single resource with high performance [2]. Grid computing allows a group of computers to share the system securely and optimizes their collective resources to meet required workloads by using open standards OGSA (Open Grid Services Architecture) [3]. The Grid allows executing jobs in different nodes. In order to perform job scheduling and resource management at Grid level, usually it has used a Resource scheduler or a meta-scheduler. A scheduler is fundamental in any large-scale Grid environment. The task of a Grid resource scheduler is to dynamically identify and characterize the available resources, and to select the most appropriate resources in order to submit jobs. In grid scheduling discussion, selecting best nodes with looking at economic and fault tolerance criteria is considerable [4]. Choosing the suitable fault tolerance resource for a user job to meet predefined constraints such as deadline, speedup and cost of execution is an important problem in the grids. In our approach, we highly have solved some of these problems.

As known, grid scheduling consists of three steps. The first step is resource discovery and filtering, and the second is selecting nodes and scheduling jobs to related nodes, and the last step is submitting and monitoring jobs. Surely, step 2 is vital because some nodes always have best behavior, while some others often have fault with low performance [5].

In scheduling phase on grid, schedulers usually use some information about resources' attributes (CPU speed and load, memory) to do the scheduling. The information used by the schedulers is usually provided by an information service that is responsible for gathering data about all resources that compose the grid. Key problem is that information obtained from the grid information service (GIS) may be out of date by the time the scheduler needs it to schedule tasks. In our approach, we have used an online scheduling with novel information.

Since optimized case-based Reasoning (OCBR) is one of the preferred problem-solving strategies and machine learning techniques in complex and dynamically changing situations, we used an optimal fault tolerance approach by applying an optimized case-based Reasoning algorithm to prediction, detection and recovery of faults in grid [6].

The rest of this paper is organized as follows. Section II gives an overview on previous research in fault tolerance resource scheduling. Section III describes an optimized CBR that is used in our method. Section IV discusses the system design and implementation details of our Grid resource scheduling respectively. Section V describes experimental results and Section VI concludes the paper.

II. Related Works

In recent years, many researchers have offered many methods, frameworks or algorithms for dynamic job scheduling in different notions. The most objective was failure nodes problem in task scheduling. Unfortunately in the grid, the possibility on failure in resource node is not deniable. In the past, many fault tolerance job scheduling has been suggested for grid or cluster computing [7], [8].

S. Baghavathi Priya and et al. presented a fault tolerance approach for Task Scheduling by using Genetic algorithm on grid [5]. They have used checkpoint technique in their method that is a general-purpose method for providing fault tolerance in distributed systems. Check pointing allows the recovery to a previous correct state. Due to simplicity and understandability, their method is a good one. Nevertheless, for a large-scale computing with thousands nodes, surely this algorithm take much time. In addition, they have considered some assumptions for scheduler, which require more investigation.

GRIDTS is another infrastructure for Fault-Tolerant Scheduling in Grid environment [7]. This proposed approach allows scheduling decisions to be made with up-to-date

Manuscript received March 4, 2013; revised May 18, 2013.

The author is with the Faculty of Information Technology and Computer Engineering, Azarbaijan Shahid Madani University, Tabriz, Iran (e-mail: a.bouyer@azaruniv.edu).

information about the resources. GRIDTS provides fault-tolerant scheduling by combining a set of fault tolerance techniques to deal with crash faults in components of the system. Fault tolerance in GRIDTS is enforced using a combination of mechanisms. Their approach does not use GIS information, because GIS information might be abrogated.

B. Nazir and T. Khan present another approach that is simply performable [8]. They developed a new method for fault tolerant job scheduling in grid. Proposed approach maintains history of the fault occurrence of resource in Grid Information Service (GIS). Whenever a resource broker has job to schedule, it uses the fault occurrence from history information, and depending on this information, it uses different intensity of check pointing and replication while scheduling the job on resources that have different tendency towards fault. They claimed that it increases the percentage of jobs executed within specified deadline and allotted budget, hence helping in making grid trustworthy. However it is possible that this algorithm cannot be optimal, because data in GIS might be old and abrogated.

There are some other related works such as [9]-[11] that we devolve them to readers.

III. OCBR Algorithm

The proposed approach in this paper uses a prediction algorithm to detect the treatment of nodes for new jobs. This approach applies OCBR algorithm [6], the CBR method on the basis of Decision Tree, in order to select suitable sampling. OCBR consists of two phases:

Phase 1: primary processing of information to make Decision Tree and assigning each existing record (or sample) to its related class. Decision Tree in this research is used to select a best training set in appropriate branches, that coming job exists in those branches, for CBR algorithm [12].

Phase 2: final processing and predicting the situation of a record (coming job) by using neighboring records.

At the first step, to identify the main and efficient parameters in existing database system and to clarify their effect on final result, we do a processing operation on the obtained results. Then we try to classify the information into different classes (by using decision tree classifier). At the second step, we try to insert the desired record into its class according to previously done classification in Decision Tree. Then considering the number of desired neighbors, we select the existing records that are similar to our desired record and perform the predicting operation (CBR algorithm).

At first, the information or primary system parameters are identified and integrated. Next, we can get the final result by performing the final processing among the desired record and its neighbors (in the same class).

IV. System Architecture

In this approach, there is a local database for every node in grid that is considered to store some useful information about submitted jobs and status of execution. When a new job is submitted on a node, then a new recode will be inserted to its database. In addition, we have considered a queue on grid scheduler that is called Reservation Queue, to support faulted nodes. When one of the nodes in site is failed, if reservation queue is not empty, then a new node replace with faulted node. Scheduler is responsible to this change node, because that is aware from status of nodes in site by using message passing. We have shown a general architecture for this approach (Fig. 1). For the nonce, we have provided an isolated application that must be installed and executed on each node as well as another application that will be installed on scheduling machine for this purpose.

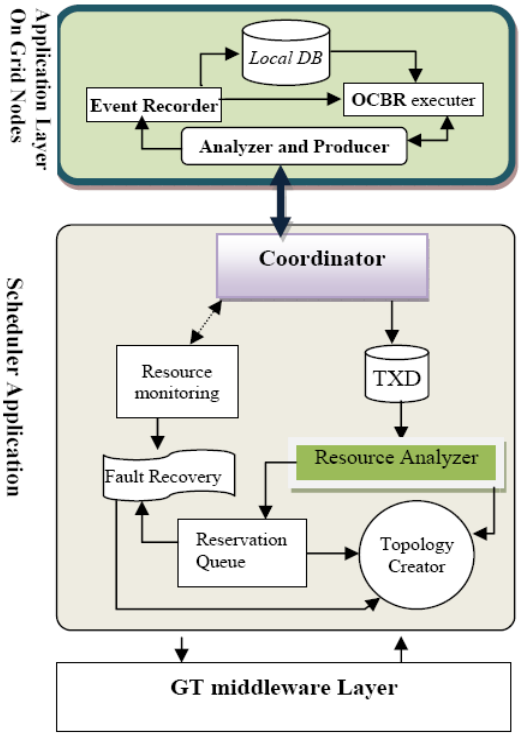

Fig. 1. The architecture of proposed approach for fault reducing.

A. Scheduler's Application

This application is responsible to select fault tolerance nodes and doing a fault tolerant cycle to retain system in a tolerant safe state. For implementing a fault tolerance scheduling, we shall do following phases:

Phase 1: coordinator is responsible to send a packet to every node on grid. This packet include some information about new coming job and a request for executing OCBR [6] in local database by desired node based-on this coming job. Scheduler wants to find nodes with the least fault and best performance in past. Then, the produced results will be sent to coordinator to be saved in a temporary XML database (TXD).

Phase 2: coordinator will insert all received results from each node in a TXD. After that, Resource Analyzer will analyze this results based-on fault tolerant criteria and job condition by executing several queries to get better and squarer decision. For example, it finds a node that has best fault tolerance for small jobs but it is not suitable for heavy jobs. Therefore, the scheduler must consider coming job status in its decision or selection. Then, Resource Analyzer

selects the best fault tolerance nodes to start operations; and extra nodes will reserve in a queue that we called it Reservation Queue. Also for each node it will consider a priority.

Phase 3: creating a Virtual topology in a Ring form, based on the nearest neighbor on the right side of each node -as near as possible and without any node repetition. Now, if there is a fault happens in one of the nodes, Fault Recovery is called to resolve it. At the beginning, Fault Recovery will check Reservation Queue to find a node for replacing with failed node, but "what will it do if the queue is empty?"

Since the job belonging to failed nodes may be an important job with high priority, so this job should not be left unfinished and scheduler is forced to run it. Therefore, Fault Recovery section will send a message to left-hand of faulted node in order to create a connection with right-hand of faulted node and its job transfer to right-hand node to continue this unfinished job there. Of course, we do not want to apply many nodes for reservation, because, we have developed this approach based on economic-based grid. As you know, in the real world, nobody likes others to use his\her computer free.

B. The Applied Virtual Topology

When scheduler had selected desired nodes, it will create a virtual topology in the form of a ring. An important point that must be mentioned is that in this topology, each node is communicating with right side node and these two nodes are the nearest possible neighbors. In other words, all nodes have a near neighbor on their right-hand. If a node failed, with the assumption that queue is empty, scheduler will do as follows:

- Step1: Fault Recovery Section scans cause of failure. If the related node has a hardware problem it will do step2. If the node is alive, but it cannot continue the its related job for the reason such as CPU-Idle RAM or Secondary Storages problem, then Fault Recovery Section will define a random time, called chance time to revival, in order to revive probability. If in this allocated time, the desired node has started again, it needn't do any operation. Otherwise, it must start step 2.
- Step2: At this step, Fault Recovery Section will transfer the job of failed node to right-hand node. If transferred job has a real-time priority, then desired node can stop its job and start this real-time job, but its job also had a Real-time priority then it can transfer previous job to next node. For getting a better performance we can use checkpoint technique in while of executing job.

Since the discovered node in grid is superabundant, therefore, we usually will not have the lack of computing resources problem. Nevertheless, it is likely that the problem pay fee for powerful and reliable resources such as Super-computers or Cluster-computers have been existed. But, surely we don't have problem in using a usual resource computing, for example Personal computers (PCs).

C. Node's Application

For better prediction, we have provided a specific Node's Application (NA) for every node in a purposed grid. NA contains an internal local database that considered for recording all events about submitted jobs by grid and only Event Recorder section can write in database. When a new job accepted and submitted on node, or while a job are completed or failed, this section will record all of mentioned events in its database. This section is more important.

Also, there is a section that we called 'Analyzer and Producer'. As mentioned above, before determining fault tolerant nodes, scheduler sends a packet that includes information about coming job (e.g. IP Sender, Size of the job, Size of needed RAM and HDD, average time needed for execution, approximate execution start time, minimum power to CPU, etc.) to each discovered nodes. When Analyzer and Producer section have received the packet, at the first time, it sends job information to event recorder to register on database; after that, if this request was acceptable according to existing resources on node, then OCBR executer will start to carry out case-based reasoning algorithm. Moreover, each node has computed the fowling measures in order to send to scheduler.

1) Average Hit Ratio (HR): This attribute represents an average rate of success in all previous times.
2) Hit Ratio for the last twenty jobs (RT).
3) Hit Ratio for this time-period on previous days (TP) based-on a estimation on coming job execution. For example, how many jobs have been executed from 1.30 AM to 2.00 AM?

The following observations are considered:

- *Count(SJ_i):* returned the number of successfully finished jobs on node$_i$
- *Count(FJ_i):* returned the number of failed jobs on node$_i$
- *Count(ST_i):* returned the number of successfully finished jobs in the last 20 submitted jobs on node$_i$
- *Count(APJ_i)*: returned the number of all jobs submitted at this time, on the previous days on node$_i$
- *Count(NSP_i):* returned the number of all jobs successfully finished at this time, on the previous days on node$_i$

$$HR_i = \frac{Count(SJ_i)}{Count(SJ_i) + Count(FJ_i)} \quad (1)$$

$$RT_i = Count(ST_i)/20$$

$$TP_i = \frac{(Count(NSPi)) \times (Count(SJi) + Count(FJ_i))}{Count(APJ_i) \times (Count(SJi) + Count(FJi)) - Count(APJ_i)^2 + 1} \quad (2)$$

$$IDLE_i = \frac{(CPU\ IDLE_i)}{100}$$

$$RAM_i = \frac{(Free\ Available\ phisical\ memory)_i}{Size\ of\ RAM_i} \quad (3)$$

Then, this above computed result along with produced rules will send to scheduler in a XML document. When Resource Analyzer take the above produced result, it computes the below formula for PR_i according to weight allocation to each parameter; of course, this weight identified after several experience.

$$PR_i = \frac{\begin{pmatrix} 0.5\times HR_i + 0.4\times RT_i + 0.8\times TP_i \\ +0.4\times IDLE_i + 0.2\times RAM_i \end{pmatrix} \times count(SJ_i)}{count(FJ_i)} \quad (4)$$

The PR_i parameter shows a status of node ith in previous history. For example, we have three nodes that their conditions almost are same but PR_i for each node are different. Whenever each node's PR_i is high then its priority also is better than others.

V. Experiment Results and Discussion

To evaluate proposed scheduler technique, we simulate a grid environment. Although we had previously provided 64 nodes in different places, this had been done for other purposes; anyway, these samples were sufficient for this research. Therefore, we used these samples in our simulator. In this simulator we considered 64 nodes. Each simulated node had its own local database. As we mentioned before, in these local database all events about job submission on the node have been saved. In this environment, there are two level schedulers; high-level scheduler and local scheduler. In this research, we have concentrated on local scheduler because each scheduler can act independently from other local schedulers. Every scheduler in desired site is scheduling nodes independently from other scheduler. Suppose that all 64 nodes are available in simulated grid and we want to select some nodes for our purposes. When we want to start execution, at first, all 64 nodes will execute OCBR algorithm and then will send obtained results to TXD database. Then scheduler's application compares all gathered results to select the best fault tolerance. These results are produced by scheduler according to priority in Table I.

TABLE I: Provider Nodes' Specifications with Their Priority

Node's Name	priority	prediction	PR_i	CPU-idle	Free RAM	Small job	Med job	Larg job
$Node_1$	0.89	0.91	24.1	0.54	0.02	VG	G	G
$Node_2$	0.87	0.95	29.8	0.77	0.04	VG	M	M
$Node_3$	0.84	0.82	21.9	0.97	0.05	G	M	M
$Node_4$	0.94	0.95	62.3	0.87	0.43	VG	VG	G
$Node_5$	0.95	0.96	75.6	0.98	0.39	EX	VG	VG
$Node_6$	0.92	0.95	56.4	0.46	0.38	VG	VG	G
$Node_7$	0.90	0.93	42.5	0.8	0.28	G	G	M
$Node_8$	0.78	0.80	20.85	0.65	1	G	M	VW
$Node_9$	0.92	0.91	48.9	0.8	0.28	VG	G	G
$Node_{10}$	0.83	0.89	34.85	0.65	1	G	G	M
…	…	…	…	…	…	…	…	…

EX= Excellent [95% to 100%];
VG=Very Good [85% to 95%]
G= Good [70% to 85%)
M= Medium [50% to 70%)
W= Weak [30% to 50%)
VW= Very Weak [0 to 30%)

Then we chose some nodes that had the best condition to execute an example job (memory need= 10.41and estimation time to execution= 850sec). After 14 experiments we reached objective results and then compared these results with another approach in [8]. The results have been represented in Fig. 2.

As you see, due to apply OCBR methods in resources scheduling, we obtained better results than FTGS method [8]. Moreover, we compared successfully finished jobs in any way in these methods that results have been showed in Fig. 3. It seems that, the New Approach tries to complete a job in one of the three existing ways. Therefore it has a good ability in successfully finishing job and it has a better fault tolerance feature.

We have brought time execution in our approach in Fig. 4 to compare with Fig. 5 that show FTGS method results. To reach this aim, we have done three experiments for small, medium, and large jobs. The results are mentioned below.

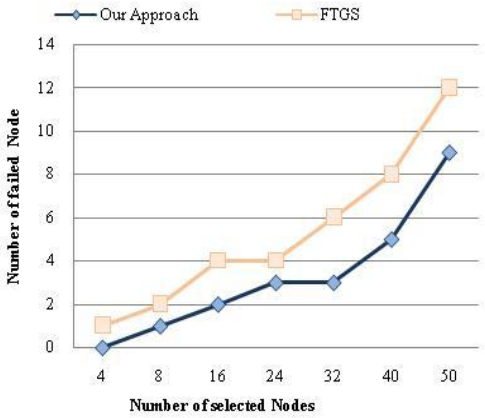

Fig. 2. The number of failure nodes in our approach and FTGS method

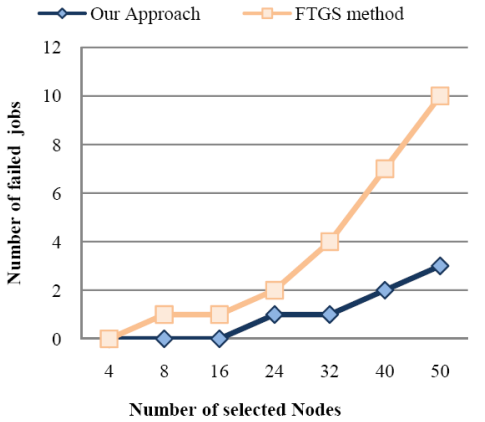

Fig. 3. Evaluation the failure jobs in our approach with FTGS method.

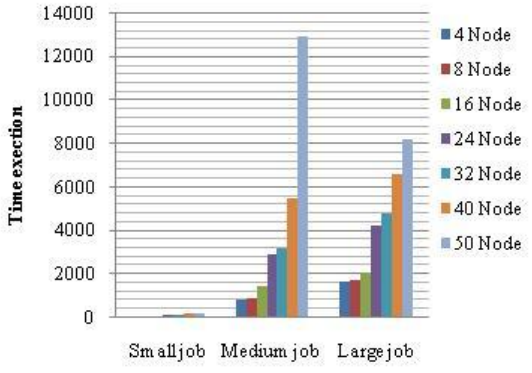

Fig. 4. Time execution for different jobs based on selected nodes by new proposed method

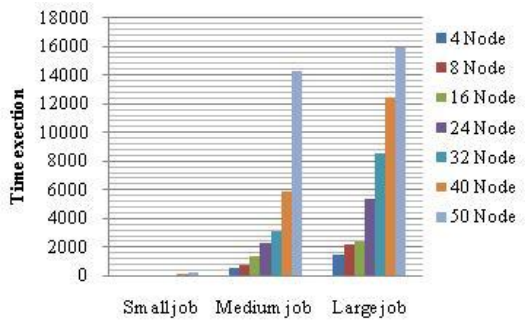

Fig. 5. Time execution for different jobs on selected nodes in FTGS method

VI. Conclusion and Future Works

In Grid environments, job failures can occur for various reasons such as resource failure, network failure, RAM or Storage or CPU Problems, and software problems. In this paper, we presented a new approach to improve grid utilization, fault tolerance in grid scheduling and decrease completion time of jobs. The proposed grid-scheduling approach can select the best fault tolerance nodes and also detect a failure node and simply manage it by using one of the provided strategies. The obtained results by our simulation indicate that the new approach may be very effective for adaptive grid scheduling due to reliability, fault tolerance, and then decrease of job completion time. As part of our future work, we are going to extend our approach by dynamic and intelligent component in scheduling phase.

References

[1] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: enabling scalable virtual organizations," *International Journal of Supercomputer Applications*, vol. 15 pp. 200–222, 2001.

[2] A. Bouyer, A. H. Abdullah, and M. H. Mokhtari, "Localized job scheduling system using cooperative and system-centric scheduling policy for market-oriented grids," *Journal of Scientific Research and Essays*, vol. 6, pp. 3729-3750, 2011.

[3] I. Foster and C. Kesselman, *The Grid 2: Blueprint for a New Computing Infrastructure*: USA: Morgan Kaufmann, 2003, ch. 17.

[4] A. Bouyer and S. M. Mousavi, "A Predictive Approach for Selecting Suitable Computing Nodes in Grid Environment by Using Data Mining Technique," in *Proc. Advances in Computational Science and Engineering*, China, 2009, pp. 190-205.

[5] S. B. Priya, M. Prakash, and K. K. Dhawan, "Fault tolerance-genetic algorithm for grid task scheduling using check point," in *Proc. the Sixth International Conference on Grid and Cooperative Computing*, Urumchi, Xinjiang, China, 16-18 Aug., 2007.

[6] A. Bouyer *et al*., "A new hybrid model using case-based reasoning and decision tree methods for improving speedup and accuracy," presented at the 4th Iadis International Conference Applied Computing 2007, Salamanca, Spain, 2007.

[7] F. Favarim *et al*., "GRIDTS: A new approach for fault-tolerant scheduling in grid computing," presented at the Network Computing and Applications (NCA 2007), Sixth IEEE International Symposium on, Cambridge, MA , 2007.

[8] B. Nazir and T. Khan, "Fault tolerant job scheduling in computational grid," presented at the Emerging Technologies, 2006. ICET '06. International Conference on, Peshawar, Pakistan, 2006.

[9] R. P. R. Duan and T. Fahringer, "Short paper: data mining-based fault prediction and detection on the grid," presented at the 15th IEEE International Symposium on High Performance Distributed Computing, Paris, Frans, 2006

[10] L.-O. Burchard *et al*., "VRM: a failure-aware grid resource management system," *International Journal of High Performance Computing and Networking*, vol. 5, pp. 215-226, 2008.

[11] R. F. Lopes and F. J. d. S. E. Silva, "Fault tolerance in a mobile agent based computational grid," in *Proc. the Sixth IEEE International Symposium on Cluster Computing and the Grid*, Singapore, 2006.

[12] X. Qin and W. C. Regli, "A study in applying case-based reasoning to engineering design: Mechanical bearing design," *Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 17, no. 3, pp. 235-252, 2003.

Asgarali Bouyer is an assistant professor in the faculty of computer engineering and information technology at Azarbaijan Shahid Madani University, Tabriz, Iran. He received Ph.D. degree in Computer Science from University of Technology Malaysia (UTM), in 2011. His research interests are in distributed computing (Grid and cloud computing), Data mining and mobile computing. Application areas include bioinformatics, cluster computing and data mining, etc. He has been involved in several Iran research projects.