An Efficient Fault Tolerant Job Scheduling and Load Balancing in Grid Computing

C. V. Elakkiyaa
Computer Science and Engineering, SNS College of Technology, Coimbatore, India

B. Gilead Baggio
Computer Science and Engineering, SNS College of Technology, Coimbatore, India

A. Arvind Raj
Computer Science and Engineering, SNS College of Technology, Coimbatore, India

L. Lalitha
Assistant Professor, Computer Science and Engineering
SNS College of Technology, Coimbatore, India

Abstract:
In Grid environment, the two main techniques that are most suitable to cope with the dynamic nature of the grid are load balancing and job replication. In Load balancing algorithm juxtaposes is considered the strong point is neighbour based and cluster based methods. We propose a genetic algorithm for job scheduling to address the heterogeneity of fault-tolerance mechanisms problem in a computational grid. We assume that the system supports four kinds fault-tolerance mechanisms, including the job retry, the job migration without check pointing, the job migration with check pointing, and the job replication mechanisms. Because each fault-tolerance mechanism has different requirements for the gene encoding, we also propose a new chromosome encoding approach to integrate the four kinds of mechanisms in a chromosome. The risky nature of the grid environment is also taken into account in the algorithm. The risk relationship between jobs and nodes are defined by the security demand and the trust level.

Key words: Computational grid, Fault tolerance, Job scheduling, Genetic algorithm

1. Introduction
Grid computing has emerged as the next-generation parallel and distributed computing methodology that aggregates dispersed heterogeneous resources for solving various kinds of large-scale applications in science, engineering and commerce. In large-scale grid environments, the underlying network connecting them is heterogeneous and bandwidth across resources varies from link to link. Not limited to the grid, in many of today’s disturbed computing environment (DCE), the computers are linked by a delay and the bandwidth limited communication medium that inherently inflicts tangible delays. Grid systems are classified into two categories: Compute and data grids. In computing grids, the main resources that are being managed by the resources management system is computed cycles, while in data grids the focus is to manage data distributed over geographical locations. The type of grid system, it is deployed in affects the architecture and the services provided by the resources management system. In this paper, we consider fault-tolerant scheduling using genetic algorithm and balancing, application load for a computational grid by taking into account grid architecture.

2. Related Work
Although load balancing, job scheduling, and fault tolerance are active areas of research in a grid environment, these areas have largely been and continue to be developed independent of one another each focusing on different aspects of computing. With respect to load balancing Jasna Balasangameshwara and Nedunchezian Raju [1] proposed That the load balancing with four policies and proved the load adjustment policy is considered as the best policy. Malarvizhi Nandagopal and Rhymend V Uthariaraj [2] proposed load-balancing involves assigning job to a resource proportional to its performance, thereby minimizing the response time of a job. However, there are wide varieties of issues that need to be considered for a heterogenous grid environment. Manimaran and Sivaram Murthy [3] proposed real-time-system in which the correctness of the system depends not only on the logical result of computation, but also on the time at which the results are produced. In dynamic scheduling, when a new set of tasks arrive at the system. Qin Zheng, Bharadwaj Veeravalli, Chen-Khong Tham [4] proposed that the need of efficient algorithm that can schedule both independent and dependent jobs on the computational grids so that jobs can run successfully even when processor failure occurs. Besides the need to deal with processor failure, grid applications.

Xiaomin Zhu, Xiao Qin,
Meikang Qiu [5] proposed heterogenous clusters, consisting of a diverse computers connected, by high speed networks, have been adopted as an efficient high performance computing platforms for computing-intensive.

3. Load Balancing Policies
In load balancing each resources \( c_i \) maintains \( p \) number of neighboring resources \( \text{LNSet}_i \), and cluster resources \( \text{LPSet}_i \), which the grid scheduler will use to select a neighboring resources for offloading jobs. Neighbors for each resource are formed in terms of transfer delay. For \( c_i, c_k \) is considered as its neighbor resources as long as the transfer delay between \( c_i, c_k \) is within \( \alpha \) times of the transfer.

3.1. Resource Efficiency Estimation Policy
Grid Scheduler maintains the submitted job list, namely
\[
S(c_i) = \begin{cases} 
-1 & \text{job is finished} \\
0 & \text{no job} \\
+1 & \text{job is unfinished} 
\end{cases} \tag{1}
\]
Finally the health of the resources is calculated by adding up all the scores.
Grid Scheduler maintains the efficiency level of neighbor-based and cluster-based resources.

3.2. Performance Benefit Factor
Performance benefit factor associated with a job it is based on communication cost taken by the resources.
\[
P_B = c(c_j, c_i, c_i, t) - c(c_j, c_i, c_k, t) \tag{3}
\]
Where \( c(c_j, c_i, c_i, t) \) is estimated communication cost at \( c_i \)
Where \( c(c_j, c_i, c_i, t) \) is estimated communication cost at \( c_i \)

3.3. Enhanced MIF Policy
Each resources \( c_i \) maintains the state information about other resources. The information is used to estimate the load and efficiency of other resources. It makes the site process for each resource. Mutual Information Feedback policy is that the rate of load dissemination is directly proportional to the job arrival rate.

3.4. Instantaneous Distribution Policy
When a new job arrives at a resource \( c_i \), the policy decides whether the job has to be executed locally or sent to other resources in \( \text{LPSet}_i, \text{LNSet}_i \). The decision depends on whether it can gain performance benefit if it is distributed to a resource in \( \text{LPSet}_i \) or \( \text{LNSet}_i \)

3.5. Load Adjustment Policy
The load adjustment policy for a resource \( c_i \) tries to continuously reduce load difference among \( c_i \) and its neighboring resources in \( \text{LNSet}_i \) by migrating jobs from heavily loaded resources to lightly loaded resources. The load adjustment policy is triggered whenever \( c_i \) receives load information from its neighboring resources. The load-balancing algorithm will use the most recent load status information and efficiency value to initiate a job. Taking into account that all resources are exchanging their loads in parallel and the dynamic nature of the environment, the network reaches a local optimum quickly. Thus, each resource submits the most loaded job to one of its neighbor resources which have the minimum load among all. If all its neighbors are busier than itself, no job is submitted from the current resource. If there are any jobs in the job queue waiting to be executed, the resource can submit them to a lighter neighbor resource.

Figure 1: The Pseudocode of instantaneous distribution policy

3.5. Load Adjustment Policy
The load adjustment policy for a resource \( c_i \) tries to continuously reduce load difference among \( c_i \) and its neighboring resources in \( \text{LNSet}_i \) by migrating jobs from heavily loaded resources to lightly loaded resources. The load adjustment policy is triggered whenever \( c_i \) receives load information from its neighboring resources. The load-balancing algorithm will use the most recent load status information and efficiency value to initiate a job. Taking into account that all resources are exchanging their loads in parallel and the dynamic nature of the environment, the network reaches a local optimum quickly. Thus, each resource submits the most loaded job to one of its neighbor resources which have the minimum load among all. If all its neighbors are busier than itself, no job is submitted from the current resource. If there are any jobs in the job queue waiting to be executed, the resource can submit them to a lighter neighbor resource.
4. Fault Tolerance Strategies

Fault tolerance scheduling policy fault-tolerant scheduling policy namely Minimum Replication Cost (MinRC) and Completion Time for independent and aperiodic jobs is MinRC handles both types of failures that can happen in a grid environment. If a resource shuts down manually, it sends a notice message to its backup and other neighbor and partner resources before shutting down. If a resource fails suddenly without any notice, then the following strategy is employed. We use job replication strategy in which each replica is independently scheduled in a different resource to achieve fault tolerance. The motivation for job replication is that resources in the grid environment can be highly underutilized; thus, these spare resources can be used to run job replicas. MinRC employs peer-to-peer fail-over strategy such that each resource is a backup of another neighbor or partner resource in the grid system.

4.1. Boundary Schedules

Replication cost is defined as the actual percentage of time needed to be scheduled for the backup besides all overloaded periods with existing backups. The neighbor and partner resources for the primary $c_i$. In case a tie happens, the boundary schedule which can complete earliest is selected.

4.2. Algorithm for Scheduling Backups

The boundary schedule which has minimum replication cost is chosen the boundary schedule which can complete earliest is selected. Let $t_i$ and $t_e$ denote start time and completion time of an existing schedule, respectively.
4.3. Schedule Eligibility

A schedule is eligible if it is within the time window and does not overlap with any primary schedule or nonoverloadable backup schedule. Specifically, there must not exist any primary schedule or nonover-loadable backup schedule that starts and/or ends in this schedule or contains this schedule. Let $t_{lb}$ and $t_{rb}$ denote start and completion time of overloadable backup schedule.

5. Proposed Work

GA-based job scheduling strategy for a large-scale computational grid to address the problem that all existing strategies assume that all computational nodes are protected by the same fault-tolerance mechanism. In fact, it is very common that different autonomous domains usually have different administrative policies and different computational sites have different hardware and software fault-tolerance supports. Therefore, we considered the computational grid in which each computational site supports one or two of four kinds of fault-tolerance mechanisms, including the job retry, the job migration without checkpointing, the job migration with checkpointing, and the job replication mechanisms. The scheduler will decide which kinds of fault-tolerance mechanisms will be applied to each individual job for more reliable computation and shorter makespan. To encode mixed kinds of fault-tolerance mechanisms into a single chromosome, we also proposed a new chromosome encoding approach.
Figure 6: The proposed genetic algorithm for fault-tolerance job Scheduling

- **Job retry (JRT) mechanism**  
  The JRT mechanism is the simplest fault-tolerance technique, which will re-execute the failed job from the beginning on the same computational node.

- **Job migration without checkpointing (JMG) mechanism**  
  The JMG mechanism will move the failed job to another computational node and re-execute the job from the beginning on the latter computational node.

- **Job migration with checkpointing (JCP) mechanism**  
  The JCP mechanism will record the state of the job periodically at run time. If the job fails, it is moved to another computational node and resumed the execution from the last checkpoint.

- **Job Replication (JRP) mechanism**  
  The JRP mechanism replicates a job to multiple computational nodes such that the job has higher success rate. If one of those replicas has already completed, then all other replicas should stop their execution to save the computing power.

6. Conclusion  
In particular, we present an approach for fault tolerance mechanisms as genetic algorithm, validating fault tolerance properties of each mechanism, finding the fitness value for the failed resources and try out different mechanism for failed resources. Therefore we considered the computational grid in which each computational site supports one or two of four kinds of fault-tolerance mechanisms, including the job retry, the job migration without check pointing, the job migration with check pointing, and the job replication mechanism.

7. Acknowledgment  
We take immense pleasure in expressing our humble note of gratitude to our project guide Mrs. L. Lalitha, Assistant Professor, Department of Computer Science and Engineering, SNS College of Technology, for her remarkable guidance and useful suggestions, which helped us in completing the paper.

8. References  
5. Xiaomin Zhu, Member, IEEE, Xiao Qin, Senior Member, IEEE, and Meikang Qiu, Senior Member, IEEE, “QoS-Aware Fault-Tolerant Scheduling for Real-Time Tasks on Heterogeneous Clusters”, IEEE TRANSACTIONS ON COMPUTERS, VOL. 60, NO. 6, JUNE 2011.