A Novel Fault-tolerant Task Scheduling Algorithm for Computational Grids

Jairam Naik K a, Satyanarayana N b

Research Scholar, Faculty of CSE, JNTU Hyderabad, Andhra Pradesh, India.
Professor, Department of CSE, NGITS Nagole, Hyderabad, Andhra Pradesh, India.

jairam.524@gmail.com, nsn1208@gmail.com

Abstract—A computational grid is a hardware and software infrastructure that provides consistent, dependable, pervasive and expensive access to high-end computational capabilities in a multi-institutional virtual organization. Computational grids provide computing power needed for execution of tasks. Scheduling the task in computing grid is an important problem. To select and assign the best resources for task, we need a good scheduling algorithm in grids. As grids typically consist of strongly varying and geographically distributed resources, choosing a fault-tolerant computational resource is an important issue. The main scheduling strategy of most fault-tolerant scheduling algorithms depends on the response time and fault indicator when selecting a resource to execute a task.

In this paper, a scheduling algorithm is proposed to select the resource, which depends on a new factor called Scheduling Success indicator (SSI). This factor consists of the response time, success rate and the predicted Experience of grid resources. Whenever a grid scheduler has tasks to schedule on grid resources, it uses the Scheduling Success indicator to generate the scheduling decisions. The main scheduling strategy of the Fault-tolerant algorithm is to select resources that have lowest tendency to fail and having more experience in task execution.

Extensive experiment simulations are conducted to quantify the performance of the proposed algorithm on GridSim. GridSim is a Java based discrete-event Grid simulation toolkit. Experiments have shown that the proposed algorithm can considerably improve grid performance in terms of throughput, failure tendency and worth.

Keywords—Resources, Computational Grid, Fault-tolerant, success rate, Simulation, Failure tendency, Resource experience

I. INTRODUCTION

Grid computing technology of Computer Science has emerged and evolved over the past years from the theoretical research to the application environment. Availability of powerful computers, proliferation of the Internet Technology and the high-speed networks as low-cost commodity components are changing the way we do large-scale parallel and distributed computing. The interest in coupling geographically distributed computational resources is also growing for solving large-scale problems, leading to what is popularly called the Grid and peer-to-peer (P2P) computing networks. These enable sharing, selection and aggregation of required computational and data resources for solving large-scale problems in science, engineering, and commerce.

Complexity of computational grids mainly originates from decentralized management and resource heterogeneity with different security policies. These factors often lead to an increase in the probability of resources to fail than traditional parallel and distributed systems [3] and strong variations in the grid availability. Also, as applications grow to use more resources for longer periods of time, they will inevitably encounter increasing number of resource failures [4]. This will affect the execution of the tasks assigned to the failed resources when failures occur. So, a fault-tolerant service is important in grids. Fault-tolerant is an ability of preserving the delivery of expected services by self, despite the presence of failures within the grid. The various forms of failures in grid computing systems include resource failure, network failure, and application failure [5]. Providing fault-tolerant service in a grid environment, while optimizing resource scheduling and task execution, is an important issue. In computational grids, managing the fault is a very important and challenging problem for grid application developers [5].

To detect faults and resolving them, grid applications must have fault-tolerant services. These services should enable the applications to continue their computations on the resources of the grid without terminating the applications in case of failure. These services also required to satisfy the minimum levels of quality of service (QoS) requirements for applications such as the
deadline to complete the execution, the number of computing resources, platform type, and so on. Previous research works of fault-tolerant task scheduling depend on using the fault indicator of resources in the scheduling process [6-10].

The main contribution of this paper is to introduce a fault-tolerant system with a scheduling strategy that depends on a new factor called Scheduling Success indicator (SSI). This SSI consists of the resource experience, response time and the success rate of resources in the grid. Resource experience indicates the number of tasks executed successfully by the grid resource in the given deadline. Resource experience and fault rates are the better indicator for resource failure tendency than the fault indicator. The resource with more experience and the lower failure rate will have the lower tendency to fail.

This paper is organized as follows: Section II briefs the related work for providing fault tolerant service in computational grids. In Section III, the problem is described. Section IV presents the architecture of the proposed scheduling system. Section V describes results and the performance of the system. Section VI future scope and concludes the paper.

II. RELATED WORK

Faults occur when a grid resource is unable to complete its task in the given deadline [11]. This may happens due to resource failure, task failure, network failure or selecting unpredicted resource for execution. In this paper, only the computational resource failure is considered. In general, failures may be handled either before scheduling tasks on grid resources, called proactive fault tolerant or after scheduling tasks on grid resources, called post-active fault tolerant. Post-active approaches, using techniques of task monitoring, are relatively easier to implement. This approach does not take into consideration the failure history of the resources, when taking the scheduling decision. That means regardless of its failure history, the resource with the minimum response time is selected, but there is a high probability to select resources that frequently fail. The system will apply a new scheduling decision to select for another suitable resource to execute the task, if the selected resource is unavailable. This will consume more time and thus degrade the performance of the grid.

On the other hand, proactive approaches require more information about grid resources and work in a probabilistic fashion. In proactive approaches, the decisions are made in advance of task execution to address possible failures in the grid. An effective proactive approach should consider entire information required in making the task scheduling decision, to avoid any task from any possible failures. Unlike post-active methods, where re-submission of tasks often leads to a decrease in throughput [7, 8], this approach potentially reduces the failure rates within the grid, and also increases the capacity and throughput. Post-active approach was applied by most of the researchers to deal with failures and a few of them apply the proactive approach.

In [9], Zheng and Veeravalli presented a fault-tolerant scheduling approach that schedules direct acyclic graphs (DAGs) in grids. DAG considers communication delays so that service failures can be avoided in the presence of processor faults. Their approach depends on that as tasks in a DAG have dependence on each other, a task must be scheduled to make sure that it will succeed when any of its predecessors fails due to a processor failure.

Jairam Naik et al. [1] presented a most suitable fault tolerant job scheduling system for computational grids. This strategy maintains closeness factor of grid resources. The scheduling decisions are made based on the value of the closeness factor and fault rate. In [6], Nandagopal and Uthariaraj considered minimum total time to release (MTTR) job scheduling algorithm. Also, their scheduler uses the fault index and the response time of resources when making scheduling decisions. Abawajy[10] presented a distributed fault-tolerant scheduling (DFTS) algorithm that couples job replication with job scheduling. He assumes that grid is partitioned into sites and each site has a scheduling manager to take scheduling decision. Each scheduling manager acts as a mirror for another scheduling manager. The algorithm uses fixed number of replications for each job. Each job replication is scheduled to a different site for execution. Chetepen[11]et al. suggested some scheduling heuristics based on
rescheduling of failed jobs and task replication. Their idea is suitable for scheduling any application with independent and dynamic jobs does not depend on particular grid architecture.

Reviewing literature survey that resources of computational grids are failure prone and there is a need of recommending proactive fault tolerant scheduling mechanisms to reduce the impact of resources failure over grid performance. Also there is a need of recommending experienced resource for executing tasks. Experienced resource is the one which executed more number of tasks successfully in time among the other resources of the grid.

The proactive scheduling system [2] considers the fault indicator of available resources in the grid. This fault indicator is maintained by taking into considering the failure history information of the grid resource. The resource fault rate is a better indicator for resource failure tendency. The resource with the lower failure rate will have a lower tendency to fail.

In most cases, fault indicator is not the only suitable factor for determining the resource failure history. For example, if we have two grid resources R1 and R2 and the total numbers of tasks assigned to each one are 10 and 100 respectively. R1 completes 9 and does not complete 1. R2 completes 90 and does not complete 10. R1 will have fault rate = 10% and R2 also have fault rate = 10%. Both resources have fault indicator = 10%. Which resource will be selected in this case R1 or R2? Here the experience of the grid resources is the new factor, and is required to consider the experience of the resource before scheduling the tasks on them. Hence, the fault indicator is not an effective factor in choosing the most reliable resource for executing the privileged task.

In this paper, we introduced a Scheduling Success indicator (SSI); it is a combination of success rate and resource experience. The experience of a resource is determined by considering the number of tasks executed successfully by the resource of the grid and the total number of tasks executed by the grid. This will gives the worth of a resource in the grid. The SSI is a better indicator for resource failure tendency than the fault indicator. The resource with more worth will have a lower tendency to fail. According to the above example, R1 will have worth rate = 0.81% and R2 will have worth rate = 8.1%, if the number of tasks executed by the grid are 1000. So, R2 is selected because it has the grater worth rate than R1 and, consequently, the lowest tendency to fail.

III. PROBLEM DESCRIPTION

User tasks are executed by the resources of computational grid as follows: users submit their tasks to the grid with their QoS requirements. Grid scheduler schedules these tasks on the most suitable resources according to the resource response time and the fault indicator. The resource executes the job and the result is submitted to the user. Executing jobs on grid resources using the above manner has a major drawback: while there are resources that fulfill the criterion of the response time and failure history, they have a tendency to fail. Also, fault indicator is not a suitable indicator for determining the resource worth. This results in selecting resources that may have a higher tendency to fail.

Within this scope, the main contribution of this paper is to introduce a fault-tolerant system with a scheduling strategy that depends on a new factor called Scheduling Success Indicator (SSI). The main idea behind the proposed system is to avoid resources that frequently fail and select the resource with more worth rate. The proposed system is compared with the scheduling system presented in [2] that mainly depends on the response time and fault indicator.

IV. FAULT-TOLERANT SCHEDULING SYSTEM

The main strategy of the proposed scheduling system depends on the response time, worth rate and experience in selecting a resource for executing a certain task. Also, the system uses event-handler to detect and handle resource failures in the grid.

A. Components of the proposed System

The interaction between the components of the proposed system is shown in Fig. 1. The system depends on using event-handler to transfer tasks, results, and scheduling information to resources of the grid. The grater success rate of a resource indicates a higher probability of availability. The proposed system has five main components:
Grid portal, scheduler, Resource information server (RIS), fault handler, and the grid resources. The grid portal provides an interface to users to submit their tasks for execution. The scheduler selects the optimal and suitable resources to execute the task. The scheduling decisions of the scheduler are based on the response time, experience and the Success rate of the grid resources.

The Resource information server (RIS) contains information about all resources in the grid. The information can include computation speed, memory space available, work load, success rate and so on. The RIS supplies the scheduler with the required information. The fault handler is responsible for detecting faults of resources and estimating the rate of resources failure or success.

B. The scheduler’s Operation

Users can submit their tasks through the grid portal. The scheduler receives user tasks and its information from the grid portal. Task information includes task number, task type, and task size. It also receives QoS requirements of each task such as the deadline to complete its execution, the number of required grid resources and the type of these resources. The scheduler assigns each task to the most reliable, suitable, and available resource to execute the task. The most reliable resource is the resource that has a higher success rate. This can be known from the history of the resource success or failures stored in the RIS. RIS provides, the fault and success rate of each resource in the grid is stored.

The success rate \( P_{sj} \), fault rate \( p_{fj} \), Experience \( E_j \), and Worth \( W_j \) of resource \( j \) is defined by:

\[
P_{sj} = \frac{Ns}{Ns + Nf} \quad \text{------- (1)}
\]

\[
p_{fj} = \frac{Nf}{Ns + Nf} \quad \text{------- (2)}
\]

\[
E_j = \frac{Nf}{Gs} \quad \text{------- (3)}
\]

\[
W_j = (1 - p_{fj}) * E_j \quad \text{------- (4)}
\]

Where \( N_s \) is the number of times the resource has completed tasks successfully in time and \( N_f \) is the number of times the resource has failed to complete the tasks assigned. Each time a resource fails to complete a task increases the value of \( N_f \) by 1 and the task assigned to that resource will be distributed to other suitable resources in the grid. Otherwise, the value of the \( N_s \) is increased by 1. The value of \( P_{sj} \) or \( P_{sj} \) is used by the scheduler when taking scheduling decisions. The most reliable resource will be the resource with the minimum value of \( P_{fj} \) or maximum value of \( P_{sj} \). \( G_s \) is the total number of tasks successfully executed by the Grid. \( G_s \) is the summation of the tasks executed by all the resources of the Grid.

To achieve its purpose, the scheduler creates a two-dimensional matrix named SSI matrix. Each entry in the matrix represents the successful scheduling indicator for each task on each suitable resource in the grid. Assuming there are \( m \) resources and \( n \) tasks, the SSI matrix will be as follows:

\[
\begin{bmatrix}
R_1 & SSI_{11} & SSI_{12} & \ldots & SSI_{1m} \\
R_2 & SSI_{21} & SSI_{22} & \ldots & SSI_{2m} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
R_m & SSI_{m1} & SSI_{m2} & \ldots & SSI_{mm}
\end{bmatrix}
\]

Where \( SSI_{ij} \) is the scheduling success indicator of task \( i \) when assigned to resource \( j \). The scheduling success indicator combines the response time of the resource of the submitted task, the success rate and the experience of that resource. The response time is the summation of the task transmission time from the scheduler to the resource on which the task will be executed, the task execution time on that resource and the transmission time of task’s execution results from the resource to the scheduler. The response time of a resource \( j \) for a task \( i \) is defined as:
\[ T_{tj} = T_{tj} + T_{ej} + T_{rj} \quad ------- \quad (5) \]

Where \( T_{tj} \) is the task's transmission time from the scheduler to the resource \( j \), \( T_{ej} \) is the task's execution time on the resource \( j \), and \( T_{rj} \) is the time for transferring results from the resource \( j \) to the scheduler. \( T_{tj} \), \( T_{ej} \), \( T_{rj} \) can be defined by:

\[ T_{tj} = \frac{K_i}{BW_j} \quad T_{ej} = \frac{T_i}{R_{sj}} \quad T_{rj} = \frac{K_{ir}}{BW_j} \]

Where \( K_i \) is the size of a given task \( i \) and \( BW_j \) is the bandwidth between the Grid Scheduler and the resource \( j \) on which the task \( i \) can be executed. \( T_{ej} \) is defined as above. Where \( T_i \) is the required execution time for task \( i \) and \( R_{sj} \) is the speed of the resource \( j \). The value of \( T_{rj} \) depends on the size of results obtained after executing the task is defined above. Where \( K_{ir} \) is the size of results obtained after executing task \( i \). Thus, the scheduling success indicator of resource \( j \) when executing task \( i \) is defined by:

\[ SSI_{ij} = T_{tj} \times (100 - W_j) \quad ------- \quad (6) \]

Each row in the SSI matrix is sorted in an ascending order according to the scheduling indicator of each resource. The scheduler gives the list of resources selected to execute each task to the task Carrier (TC). For example, the list that will be assigned to task \( j_1 \) will be as follows:

\[ R_{j1} = [R_1 \quad R_2 \quad \ldots \quad R_m] \]

**C. The Carrier’s Role**

As grid resources are geographically distributed, the carrier’s role for fault tolerant issues in computational grids, tracing the states of resources, tasks, and network components is a big challenge. Carrier Agents provide mobility and intelligence which can be exploited to face this challenge. The proposed system comprises of the following carries:

1. **Scheduler carrier (SC):** It resides in the scheduler, receives task information and the user QoS from the grid portal. The carrier computes the two-dimensional matrix that contains the success scheduling indicators of the resources. This communicates with the task carrier and gives it its assigned tasks and the list of resources that each task can be executed on. This list is extracted from the composed SSI matrix. The SC sorts this list in an ascending order according to the value of the Scheduling Success indicator of each resource.

2. **Task Carrier (TC):** This carries out tasks to be executed with a sorted list for each task that contains resources scheduled for it. The TC receives this list from the SC. The carrier is dispatched from the scheduler. It traces the grid and distributes tasks to the scheduled resources.

The carrier migrates to the first resource in the resources list of each task. If this resource is down the carrier will go to the next resource in the list. The carrier will continue do this until it worthful resource and delivers the task to it. During its trip, the TC saves a list of the worthless resources. Upon completing its trip in the grid, the carrier migrates to the fault handler in order to update the fault rate values of these resources.

3. **Result Carrier (RC):** There is a one RC for each TC. The main function of this carrier is to wander through the grid to collect results of the successfully completed tasks. The RC visits only resources that have tasks to execute.

The carrier gets results of tasks executed on these resources and returns it to the scheduler. This carrier will start migration after the expected execution time of the first task in the list. The scheduler can now provide users with results.

4. **Fault handler carrier (FHC):** It resides in the fault handler. This carrier keeps track of the fault rates, success rates, and experience and worth rate of the resources in the grid. It performs that through notifications received from the TC. This list contains a mark for each resource. Resources that successfully complete their assigned tasks are marked as success and other resources are marked as failed. Then, the carrier updates the history of all resources in the list and estimates the value of the resource’s fault rate and success rate.

**V. PERFORMANCE ANALYSIS**

Our experiments are done on GridSim, a Java based grid simulators. In order to carry out the present study, a grid simulator is composed using java 7 platform. The simulator supports modelling and simulation of applications, grid resources and user. It enables the creation of application tasks and mapping of tasks to resources and provides fault-tolerant services by allowing injection of faults in the grid. To measure the performance of our systems following two metrics are considered.
A. Throughput

In measuring the performance of fault tolerant systems, throughput is used as one of the standard and important metrics. Throughput of a system is defined as:

\[
\text{Throughput} (n) = \frac{n}{T_n}
\]

Where \( n \), \( T_n \) are the total numbers of tasks submitted, total amount of time required to complete \( n \) task respectively. Throughput is used for measuring the ability of the grid to accommodate tasks. Fig. 2 show the throughput comparison of the FI based and SSI based task scheduling systems for number of tasks submitted. The number of tasks is 500, 1000, 1500, and 2000. We observed the percentage of throughput is increased from 70% to 90%. In our experiment, the percentage assumed is from 10% to 50%.

![Fig. 2. Throughput for number of Tasks=500](image)

B. Failure tendency

The second metric we have introduced in this work is Failure Tendency. It is the percentage of the tendency of grid resources to fail and is defined as:

\[
\text{Failure Tendency} = \sum_{m} \left( \frac{PF}{m} \right) \times 100\%
\]

Where \( m \) is the total number of grid resource. Faulty behaviour of the system is expected with this metric. It gives us a failure prediction about the grid resources when using a certain scheduling system. The Failure Tendency of our system is compared with that of the other system with different number of tasks submitted and with grids having 500 and 1000 resources. The comparison is shown in Fig.3. The figure shows that the proposed system has a lower Failure Tendency than the other system.

![Fig. 3. Failure Tendency for number of Grid Resources=500](image)

VI. CONCLUSION

In this paper, a fault-tolerant task scheduling system for grids is proposed and presented. The performance of the system is examined under different conditions with FI based system that depends on the response time and the fault indicator [2] when selecting the resources. The metrics used for evaluation are throughput, and failure tendency. It is observed that the throughput of our system is better than the FI based system. Also, the proposed system improves the success rate and the availability when compared with the FI based system. Thus, it can be concluded that under higher worth rate and lower failure tendency the proposed system provides better performance over the FI based system.

REFERENCES