This chapter presents a Trust Oriented Cooperative Resource Scheduling (TOCRS) strategy in grid computing. This strategy integrates the proactive and post-active failure handling approaches to reduce the probability of failure at the first instance and to reduce the impact of failure on task execution at the second instance. Resources are selected based on their trust in grid to organize the Cooperative Computing System (CCS). Resource trust expresses its commitment for service. Therefore, the most trusted resources are selected to organize the CCS. TOCRS improves the reliability and performance of task execution in comparison to CRS presented in Chapter 3 and Trust Oriented Resource Scheduling (TORS) approaches proposed in literature.

4.1 Background

4.1.1 Service Market Analogy of Grid Computing

Grid computing has been transformed into a computational market where computing power, storage, software and other services are rented to solve the computing problems (Buuya and Vazhkudai, 2001). Thus, grid may be perceived as a service-oriented computing paradigm, which includes service providers, service consumers and their interaction models. The rationale of service-oriented computing paradigm is the convergence of Service Oriented Architecture (SOA) and grid (Wahib et al., 2008, Smith et al., 2004). SOA is developed on top of web service standards such as WSDL.
(Christensen et al., 2001) and SOAP (WWW Consortium, 2003). With the integration of SOA and grid, both hardware and software resources are being presented as abstract services to be accessed through standard web service protocols. Many resource pricing models and market driven resource management policies have been suggested by Broberg et al. (2008) to consolidate the market orientation of grid.

Eymann et al. (2008) comprehended the similarity of grid market with the other imperfect markets, like a second-hand car market. Real market has the attributes of service advertizing, trading, negotiation, discovery and different economic models to setup the relationships between prospective service providers and consumers. Analogous to the real market, in grid environment, the functions of resource advertizing, resource trading (Schnizler et al., 2008, Izakian et al., 2010, Vouros et al., 2010), service negotiation (Adabi et al., 2013) resource discovery (Moreno-Vozmediano, 2009, Ebadi and Khanli, 2011), specification of service level agreements (Litke et al., 2008) and various economic models (Abramson et al., 2002, Haque et al., 2011) are available to establish and guide the service oriented relationships among resource (service) providers and consumers. Therefore, service oriented grid provides a wide range of resource usage options to the consumers.

4.1.2 Behavior driven Relationships among Real/Grid Market Entities

Future service relationships among service providers and consumers in the real market are greatly influenced from their past behavior. Market entities may set the mutual preferences based on their past behavior (performance). The idea of using the past behavior of resources to decide their future schedule has been extensively explored in
grid computing also. The following approaches are primarily used in grid computing to represent and model the past resource behavior. Future behavior inferences are made from the past behavior of resources to finalize the schedule.

- Resource availability modeling
- Resource performance monitoring
- Modeling of resource trust/reputation

These are the proactive approaches which conceptualize the strategy of right resource selection while making the scheduling decisions in grid. The resources in grid computing having the highest probability of availability, reliability and performance in the past are selected during scheduling. The resources with high availability and performance in the past as well as having high trust/reputation are expected to continue the same characteristics in near future also. Therefore, the availability and trust/reputation of the resources is modeled as well as their past performance is monitored to predict their future performance, availability and reliability.

4.1.3 Catastrophic Failures in Grid

Grid is subjected to catastrophic failures (Valcarenghi et al., 2008). A rightly selected resource may not be able to deliver the service reliability and performance as expected if it gets disconnected due to a network failure. Therefore, the proactive resource scheduling approaches must be integrated with post-active resource scheduling approaches to ensure the reliability and performance during an unexpected or catastrophic failure in grid. In the presented TOCRS strategy, the features of proactive
and post-active resource scheduling approaches are integrated. Cooperative computing is implemented on top of the proactive resource scheduling approach. Any one of the proactive resource scheduling approaches mentioned in Section 4.1.1 may be used beneath cooperative computing in grid. For experimentation purpose, in the work, the modeling of trust and reputation of the resources is used to propose TOCRS strategy in grid computing. The proposed scheduling strategy improves the service reliability and performance of CRS strategy proposed in Chapter 3 and Trust Oriented Resource Scheduling (TORS) in grid used in literature.

4.1.4 Grid Computing Environment

Figure 4.1 presents the considered grid environment. Each resource has a trust parameter, which reflects its trust and reputation in grid environment. Resource Trust

![Diagram of Grid Computing Environment]

Figure 4.1: The considered trust oriented grid environment
Evaluator (RTE) is used to evaluate and update the trust parameter of resources regularly at the end of each service transaction with them. Resource trust may increase or decrease over the period based on the service provided by it. Resource Information Server (RIS) stores the information about registered grid resources including their trust parameter.

4.2 Modeling of Resource Trust in Grid Computing

Trust has been defined as the firm belief in the competence of an entity to act as expected, such that this firm belief is not fixed value associated with the entity but rather it is subject to the entity’s behavior and applies only within a specific context at a given time (Azzedin and Maheswaran, 2002). Resource trust in grid environment is generated by observing their service behavior over a span of time. Trust value of the grid resources reflects their commitments for service. Therefore, if the scheduling decisions made by considering the trust value of resources in grid computing produce better results.

For the consideration of resource trust in scheduling, efficient trust modeling is required. Objective of this work is not to propose the modeling trust but to devise the trust oriented resource scheduling strategy in grid computing on top of the existing trust models. In the work, for experimentation, a simple trust model is used to evaluate and store the resource trust. A trust parameter of each resource is stored in the RIS along with other resource information. The present trust and reputation of the grid resources is represented through their trust parameter. The value of the trust parameter is generated on continuous basis in grid based on their regular service performance. Trust
parameter of a resource is updated at the end of each service transaction with it. A reliable and successful service increases the resource trust, whereas an unreliable and unsuccessful service decreases the resource trust. Trust parameter is increased as \((\text{resource\_trust} + 1)/2\) for a reliable and successful service and decreased as \((\text{resource\_trust} + 0)/2\) for an unreliable and unsuccessful service. Since the trust parameter of resources is updated dynamically at the end of each service transaction, it reflects the trust value of the resources over a span of time.

4.3 Trust Oriented Cooperative Computing System Model

In the system model, a group of most trusted resources in grid are organized together as a CCS to achieve a common objective. While participating in CCS, grid resources do not relent their autonomy. On receiving the request for task execution, resources manager perform resource matching in grid using RIS. The \(n\) resources matching to the task service specifications and having the high trusted value are selected from grid to formulate a trust oriented CCS. Execution of the computing task is dispatched to a resource with highest trust value. This resource acts as active resource in CCS, whereas rest of the resources acts as active-standby to provide execution backup to the computing task.

During the failure of a resource in CCS, its trust value is updated by the RTE. Failed resource is replaced with an alternative matching trusted resource from grid to maintain the service reliability of the CCS. Because the trusted resources are selected to execute the computing task, its probability of failure decreases. When the computing
task completes the execution, RTE updates the trust parameter of the resources deployed in its execution.

4.3.1 Assumptions of the Model

a) Resources with similar statistical characteristics are available in grid. These resources are ready to work in CCS to execute the computing task.

b) Resources in the CCS have the binary states as available and unavailable for the allocated computing task.

c) Duration of the available state of a resource is independent of the duration of finding a matching resource from grid for its replacement. These durations are exponentially distributed.

d) A resource in CCS may become unavailable (fail) due to (a) breakdown or network failure (b) resource log off from grid for a local interruption or at the owner’s will (c) switching of the resource to another high priority task in grid etc.

e) Resource manager is considered as reliable. Which implies that resource manager will not fail during the execution of the computing task allocated to CCS.

f) Execution of the computing task is delayed during task migration in the CCS due to its rollback to the recently saved checkpoint.

g) Resources are independent and have different variable failure rates. Rate of resource failure is driven from their trust and reputation in grid.

The assumptions have been commonly used in literature to model the grid environment. An explanation has already been included in Chapter 3. Assumption (g)
represents the trust-based model of resource failure and its effect on task execution. The probability of failure of a resource decreases for high trust value.

4.3.2 Reliability Evaluation of the Model

Resources included in CCS have different rate of failure. Rate of failure of each resource is derived from its trust value in grid. The mean duration of resource replacement in CCS during the failure of resources is considered to be similar and constant. Therefore, the reliability Equations (3.10) and (3.11) (Page 52) are reproduced here to represent the reliability of a resource and CCS in trust oriented grid environment.

\[
R_i(t) = \frac{\lambda_i}{\lambda_i + \mu} \exp \left[ - (\lambda_i + \mu) t \right] + \frac{\mu}{\lambda_i + \mu} \quad (4.1)
\]

\[
R_{CCS}(t) = 1 - \left[ (1 - R_1(t)) x (1 - R_2(t)) x ... x (1 - R_n(t)) \right] \quad (4.2)
\]

Resources with different rate of failure leave and join CCS during execution of the computing task. Therefore, service reliability of CCS is observed as its time average service reliability over task duration. To draw the time average service reliability of CCS, service reliability of CCS is computed at discrete intervals when a resource leaves or joins CCS during task execution through Equation (4.2).

For example, suppose a CCS is organized using three resources to execute a computing task of 35 minutes duration. If four resources are available to organize the CCS, service reliability of each resource is given as A -- 0.8, B -- 0.7, C -- 0.9, D -- 0.6.
Figure 4.2 depicts the state of CCS during execution of the computing task (number of resources present in CCS). Alphabets in the square brackets represent the resources present in CCS during a span of task duration.

\[
R(t) = \frac{10 \times [1-(1-0.8)(1-0.7)(1-0.9)] + 5 \times [1-(1-0.8)(1-0.7)] + 10 \times [1-(1-0.8)(1-0.7)(1-0.6)] + 5 \times [1-(1-0.7)(1-0.6)] + 5 \times [1-(1-0.6)]}{35} = 0.91
\]

**Figure 4.2: State of CCS during execution of the computing task**

In the figure, for first 10 minutes resources A, B and C are present in the CCS. For next 5 minutes resources A and B are present in the CCS. Similarly, different number of resources is available in the CCS during the execution of remaining task. The time average service reliability of CCS over the task duration is therefore computed as

**4.4 Simulation and Performance Evaluation**

In the simulation experiments, 1000 grid resources are considered to be registered in grid. Trust parameter of each resource is initialized between 0 and 1 using function
rand() in ‘C’. Resource trust is updated dynamically at the end of a service transaction with it by the RTE using algorithm, Update_resource_trust(). Resource selection is made using the function Locate_resource(); Pseudo code of the function used to organize the trust oriented cooperative computing system along with the other main functions used in simulation are given in Annexure - 4A. Most of the functions used to implement CRS are also used in TOCRS strategy.

500 independent computing tasks are configured for execution in grid environment. Arrival and duration of the tasks is assumed to unknown in advance and is generated through exponential function in the simulation. For evaluation of the proposed scheduling strategy extensive simulation experiments are performed for different task durations. Service reliability, average task delay and system throughput is observed in the simulation experiments. Simulation results show that TOCRS strategy improves the reliability, service delay and system throughput in comparison to the CRS strategy proposed in Chapter 3 and a Trust Oriented Resource Scheduling (TORS) (Kavitha and Sankaranarayanan, 2011, Nagarathna et al., 2012) approach proposed in literature by many researchers.

4.4.1 Service Reliability

Service reliability of the TOCRS is observed during multiple simulation experiments. A task set consisting of 500 computing tasks is executed in each simulation experiments by varying the mean task duration. Time average service reliability obtained in the experiments is compared with the service reliability of CRS strategy proposed in Chapter 3 and TORS strategy used in literature. Figure 4.3 presents the reliability
comparison. Figure shows that the reliability of TORS decreases as the size of computing task increases. In TORS, single trusted resource is selected in grid to execute the computing task. No backup resource is provided to the computing task. Therefore, execution of the task fails during a catastrophic failure in grid. For long durations the probability of catastrophic failure increases. TOCRS and CRS maintain the service reliability irrespective of the duration of computing task due to the organization of the CCS in these scheduling strategies which acquires a steady state for long duration (Page 53, Equation (3.13))

TOCRS strategy achieves better service in comparison to CRS strategy. In TOCRS strategy, resources with high trust value are included in the CCS. As a result, the probability of failure of the resources decreases which in turn decreases the probability of failure of the CCS. Therefore, the probability of failure or suspension of the computing task is quite low in TOCRS. In CRS, resources are included in CCS without considering their trust value. The resource with high failure rate may also be included in CCS, which increases the probability of its failure. As a result the probability of failure or suspension of the computing task increases. Therefore, TOCRS strategy outperforms the CRS strategy in the context of service reliability. The detailed results (service reliability) corresponding to the figures are given in Annexure - 4B.

4.4.2 System Throughput

To evaluate the system throughput, execution time of the task set is observed in different simulation experiments for different task duration. Figure 4.4 plots the execution time of the task set obtained through TORS, CRS and TOCRS strategy.
Figure 4.5 plots the corresponding throughput (per hour) for different scheduling strategies. Comparison of the throughput shows that TOCRS strategy outperforms the TORS and CRS due to the low probability of failure of the resources included in the CCS. Execution of the computing task fails only if all the resources fail in the CCS. Because of the inclusion of trusted resources in CCS, its probability of failure decreases.

The probability of task failure is highest in TORS strategy because only a single trusted resource is selected to execute the computing task. No execution backup is provided. In CRS, although active standby resources are available in CCS but the resources are selected without considering their trust value. The probability of failure of CCS increases if the resources with high rate of failure are included in CCS. As a result the TOCRS strategy achieves better throughput in comparison to CRS strategy. The detailed results (execution time of 500 computing tasks) corresponding to the figures are given in Annexure - 4B.

4.4.3 Task Execution Delay

Execution of the computing task is delayed if the computing system fails (become unavailable) during its execution. The probability of failure of the computing system using TOCRS is minimum in comparison to the TONCRS and CRS. Therefore minimum task delay is observed in TOCRS. Figure 4.6 plots the average task delay during the execution of task set through different resource scheduling approaches. Task delay is observed for different task durations. Figure shows a steep rise of the average task delay in TORS strategy. Average task delay is maintained in CRS and TOCRS.
irrespective to the duration of the computing task. Due to the selection of the most trusted resources, which has low rate of failure, TOCRS outperforms the CRS strategy proposed in Chapter 3. The detailed results (execution delay during the execution of 500 computing tasks) corresponding to the figures are given in Annexure - 4B.

![Figure 4.3: Comparison of service reliability of different resource scheduling strategies by varying the task duration](image)
Figure 4.4: Comparison of execution time of different resource scheduling strategies by varying the task duration

Figure 4.5: Comparison of system throughput of different resource scheduling strategies by varying the task duration
4.5 Chapter Summary and Conclusions

Proactive failure handling features are integrated with CRS strategy presented in Chapter 3 to devise a TOCRS strategy. Using proactive features, the probability of failures is reduced at schedule time by making the right resource selection. Resources are selected considering their trust in grid for the organization of CCS in TOCRS strategy. Therefore, the probability of failure of the CCS decreases which increases the reliability and performance of task execution in grid computing. A trust parameter is used for each resource which reflects its present trust in grid. A trust parameter is updated regularly based on the resource behavior in grid by the RTE. In CRS strategy no consideration is given to resource trust therefore the resource with high rate of failure may be included in the CCS which increases its probability of failure. As a result the reliability and performance of task execution in CRS strategy is low as
compared to TOCRS strategy. In TORS only the proactive fault handling features are used. Resource with high trust value is selected to execute the computing task at schedule time. But in absence of cooperative execution environment, execution backup is not provided. Therefore, an unexpected failure decreases the service reliability and performance of task execution. Extensive simulation experiments are performed to evaluate and compare TOCRS strategy with TORS and CRS strategy in grid. The results demonstrate that TOCRS scheduling performs better as compared to CRS scheduling.

In the next chapter execution backup is customized in the CCS to execute the mixed tasks within their permissible delay tolerance such as maximum, average or minimum. CCS is further customized to obtain priority based task execution in the next chapter.