

# FAULT TOLERANCE IN JOB SCHEDULING THROUGH FAULT MANAGEMENT FRAMEWORK USING SOA IN GRID

V. Indhumathi<sup>1</sup> and G.M. Nasira<sup>2</sup>

<sup>1</sup>Department of Computer Science, Periyar University, India  
E-mail: sabarishindhu@gmail.com

<sup>2</sup>Department of Computer Science, Chikkanna Government Arts College, India  
E-mail: nasiragm99@yahoo.com

## Abstract

*The rapid development in computing resources has enhanced the recital of computers and abridged their costs. This accessibility of low cost prevailing computers joined with the fame of the Internet and high-speed networks has led the computing surroundings to be mapped from dispersed to grid environments. Grid is a kind of dispersed system which supports the allotment and harmonized exploit of geographically dispersed and multi-owner resources, autonomously from their physical form and site, in vibrant practical organizations that carve up the similar objective of decipher large-scale applications. Thus any type of failure can happen at any point of time and job running in grid environment might fail. Therefore fault tolerance is an imperative and demanding concern in grid computing as the steadiness of individual grid resources may not be guaranteed. In order to build computational grids more effectual and consistent fault tolerant system is required. In order to accomplish the user prospect in terms of recital and competence, the Grid system desires SOA Fault Management Framework for the sharing of tasks with fault tolerance. A Fault Management Framework endeavor to pick up the response time of user's proposed applications by ensures maximal exploitation of obtainable resources. The main aim is to avert, if probable, the stipulation where some processors are congested by means of a set of tasks while others are flippantly loaded or even at leisure.*

## Keywords:

*Resource Allocation, Job Scheduling, Load Sharing Algorithm, Fault Tolerance, Grid Environment*

## 1. INTRODUCTION

In distributed job computing location, data communiqué among contribute cluster may become a most important recital hold-up [4]. However, mainstream of the client be unsuccessful to attain flush a small part of the hypothetical hustle guarantee by means of system owed to problem, at the same time as suboptimal TCP amendment, disk recital hold-up lying on the conveyance and/or acceptance ending, and server workstation restrictions. This mean that comprise elevated swiftness networks in position is significant, except not adequate [5][6]. Being capable to efficiently utilize these elevated swiftness interrelate is flattering gradually more vital to attain elevated recital various-task computing in a broadly spread surroundings. In fact, some data grids have been effectively put into practice and worn to supply a policy intended for data relocate in the set of connections and computing at the closing stage sites. It have be usually familiar to facilitate distributed set of connections be outlay effectual way to sustain statistics transfers in this category of data exhaustive submission [7][8].

Center of attention on the subject of how to stipulation an application-specific policy to facilitate for using by a client

toward scuttle a dispersed computing job, on the same time as gathering the task requirements and SLAs. To provision proposed baseline for a task, it have got to assign suitable computing and associated resources together. This type of architecture have a (logical) topology parallel with the aim of the assignment graph on behalf of that task with the exception of probably with supplementary boundaries and nodes for fault tolerance, make use of devoted but possibly possessions, in addition to are accomplished of energetic configuration to convene the dynamic necessities of the function. One of the foremost confront to survive deal with is verdict an optimal record commencing a chore graph to the set of connections to convene the application's necessity for resource network correlation [9]. Consequently, encompass a mutual trouble of task assignment development used for the statistics dispensation, and job implementation for the data transfer [10].

Further the capability for program keep back resources in move ahead, it is too necessary for scheduler to malfunction turn into proposed system economically efficiently obtain into deliberation failure as an important element of the energetic provisioning. This aptitude of improving from feasible for various serious applications, such work is reviewed, in segment 3, modeling of the Proposed as medical, financial, and homeland refuge applications.

## 2. MOTIVATION

Grid application recital is decisive in grid computing environment so to realize high performance we have to recognize the factors that can have an effect on the recital of an application and Load Balancing is one of most imperative factor which affects the overall recital of application. The Motivation is to reduce the time required to complete all jobs; and the workload is distributed evenly to all resources depending on the speed of that workload assigned to the job. The main objective is to reduce average response time and improve throughput.

## 3. LITERATURE REVIEW

The user's tasks negotiating with resource providers based on their essential Quality of Service and on the equivalent price to reach a Service Level Agreement using algorithm Quality Particle Swarm Optimization (QPSO). Resource scheduling is carries out using Application heuristics Execution meeting user deadline. It implements above algorithm for mold and optimization of resource prophecy models based on Deadline distribution and planning distribution in V. Indhumathi proposed Particle Swarm

Optimization to improve the fault tolerance in Workload execution over grid environment [1].

Major achievements includes the design and evaluation of system architecture for grid resource monitoring and prediction through Meta heuristic conditions in V. Indhumathi and G.M. Nasira proposed meta heuristics for resource Monitoring and prediction with fault tolerance in Grid Environment [2]. Develop an approach for fault tolerance based on Platform LSF, to utilize dynamic OS multi-boot to improve resource utilization. This approach combines both a heterogeneous Platform Support & advanced Self-Management with both dynamic prioritization and dynamic scheduling in G.M. Nasira and V. Indhumathi achieved fault tolerance in grid through platform LSF technique [3]. G.M. Nasira and V. Indhumathi (2012) proposed fault tolerance within grid environment using platform LSF. Handling faults depends on resource changes [4].

R.P. Ishii and R.F. de Mello took the problem of energetic scheduling of data-intensive mutiprocessor tasks. All jobs in need of some amount of CPUs and some quantity of data to requests limited storage space before starting the job. The achievement of each job conveys some benefit (utility) to the system [12].

N.N. Dang and S.B. Lim, proposed attendant virtualization that release the assortment of latest fangled potential for datacenter management, throughout the accessibility of new automation that can be subjugated to manage and observe tasks running within systems. This tender not only innovative and more supple be in charge of to the operative by means of a management cheer up, other than that more authoritative to supple manage, throughout executive software that keep the system in a preferred status in the time of altering workload and insist [13].

S. Singh and R.K. Bawa, proposed task scheduling and availability of resources provided by GIS decrease the probability of faults, execution time and increase the execution rate [18]. W.A. Elrouf et al. proposed genetic algorithm to reach the best solution faster i.e. decreasing the finishing time [19].

#### 4. PROPOSED FAULT-TOLERANCE FRAMEWORK IN GRID

The grid underneath deliberation is implicit to be a group of service providers (nodes) each building up a discrete management domain. Moreover, it is assumed that each node is proficient of handling a service request that is either submitted by a user or delegated by a peer, afforded that it hosts the necessary service with enough capacity.

##### 4.1 FAULT MANAGEMENT FRAMEWORK IN SOA

The challenging needs of the organized services are supervised through mechanisms of reservation and allocations, the running instances of the various implemented services may surpass their allocated share of resource usage. In this implementation, may be the running service occurrences are at risk to failure earlier than completion of their tasks because of unpredicted depletion of resources such as RAM or disk swap. This would lead to introduce Fault Management Framework to avoid the above said problem.

The attitude of the Fault Management Framework is to give error handling that is peripheral to SOA. The structure is engaged through guidelines distinct in XML. These guidelines are reusable over components plus preserve to hold runtime faults. On one occasion a fault is caught, the procedure describes events that can be utilized for the SOA instance such as resource failure, retry, human intervention, rethrow fault, abort, and etc.

In Fig.1, the first step is Fault Management Framework identifies faults if it is available in Service request generated by user. The second step is match the policy of appropriate fault which is defined by SOA .The third step is identifies the actions which are required to correct the fault. Based on the action, our proposed system executes the given user task without fault.

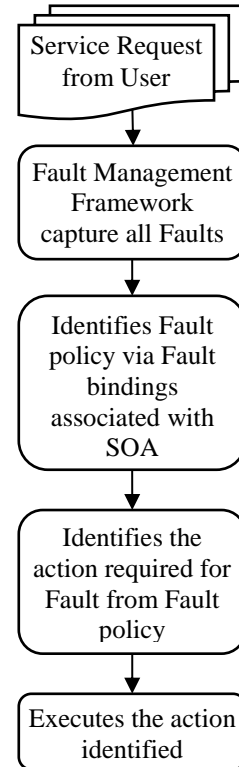


Fig.1. Proposed Fault Management Framework

##### 4.1.1 Algorithm for Workload Distribution:

**Loop**

Wait for service request from User, Collect all services as Set,

$$S = \{S_1, S_2, \dots, S_\infty\} \tag{1}$$

Apply Quick sort to the set S, it divide the service to all nodes Calculate average load of the service as:

$$Avg(s) = \frac{\sum_{n=1}^{\infty} S_n}{n} \tag{2}$$

Identify fault of each service

If (activity occurs)

Call fault policy, it catch the fault and take necessary action

Else task is executed based on workload

Iterates over all services offered by user

Return the results to user  
**End Loop**

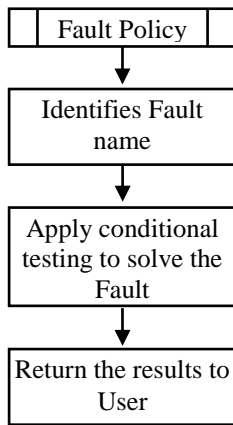


Fig.2. Fault Policies Framework

In Fig.2, the Fault policy contains more than one fault policy definitions, fault definitions and action. Fault policy identifies fault name located by different nodes. Then it calls condition element to validate faults, after that it calls action element from XML define what the action definition will do and the ids are used as references. Finally it returns result to user.

The structure of fault policies is

```

<Fault policies>
  <Fault policy>
    <Fault name>
    <Condition>
    <test>
    <Action>
      <retry>..... </retry>
    </Action>
    </test>
    </Condition>
  </Fault policy>
</Fault policies>
    
```

Fig.3. Structure of Fault policy

In Fig.3, fault policies tag contains one or more fault policy definitions, fault definitions (include conditions) and action. By using the <faultPolicy> Element, describe each and every faults connected through the policy enfolded in a <Conditions> Element. All policy name description is required for fault identification by using Query name plus a related action reference.

**4.2 RESOURCE ALLOCATION**

**Case 1: Resource Discovery**

Resource discovery involve influential which resources are accessible to a given user. At the start of this case, the set of resources is the empty set; at the end of this case the set of resources is having some values.

**Case 2: System Selection**

A single resource must be selected to schedule the job from given set of possible resources.

Table.1. Average turnaround times (sec.) for speed = (1, 5, 5, 2, 9) and load = (7, 6, 7, 6, 2)

	Total Grid	Site1	Site2	Site3	Site4	Site5
Resource Allocation	8583	2430	1816	1080	1056	67
Normal Load Sharing	3860	1943	296	983	1589	230
Practicable Load Sharing	4023	1945	299	990	1598	67

Table.2. Average turnaround times (sec.) for speed = (1, 2, 3, 4, 5) and load = (4, 4, 4, 4, 4)

	Total Grid	Site1	Site2	Site3	Site4	Site5
Resource Allocation	7832	579	5550	6789	10563	1536
Normal Load Sharing	4089	175	5210	4185	4234	339
Practicable Load Sharing	4150	175	5220	4231	4317	339

**Case 3: Job Submission**

To estimate speed intervals amongst the contributed sites we describe parameter for speed = (s1, s2, s3, s4, s5) to describe the computing speed for all five sites in grid, in which have the value of S1 is the compute speed resulting in job implementation time in the baseline workload. In addition S2 is the increasing average job execution time of the baseline workload. We define load parameter as positive real numbers are (lp1, lp2, lp3, lp4, lp5) to describe the load distribution in grid.

The Table.1 and Table.2 evaluates the effects of the possible load sharing with different speed and load in heterogeneous Grid. The Load Sharing Algorithm (LSA) is used to evaluate the practicable load Sharing. The result shows that not all sites would lead to impracticable Grid computing where some sites get tainted performance after joining in the grid.

**5. EXPERIMENTAL EVALUATION**

The makespan and Average resource utilization are the performance metrics, which measured in our experimental evaluation through the Fault Management Framework. The performance metrics such as makespan, average Resource utilization are used to show balanced resource allocation and makespan and it also defined in following sections.

**5.1 MAKESPAN**

The makespan is total amount of time required to complete group of jobs. So it is calculated by using the following formula.

$Makespan = Time\ of\ completion\ of\ last\ job - Starting\ time\ of\ the\ first\ job$

The makespan values of the various algorithms are compared with our LSA algorithm. The results show that our proposed method has minimized makespan than the other algorithms as shown in Fig.4.

**5.2 AVERAGE RESOURCE UTILIZATION**

The average resource utilization of the algorithms such as LSA, Min-min, FTMM, BSA and LBFT are shown in Fig.5 and the results shows that the proposed LSA relatively has high resource utilization.

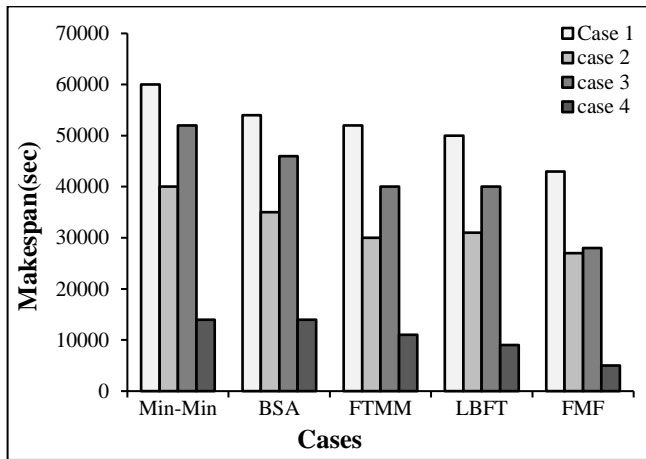


Fig.4. Makespan comparisons of various algorithms with FMF

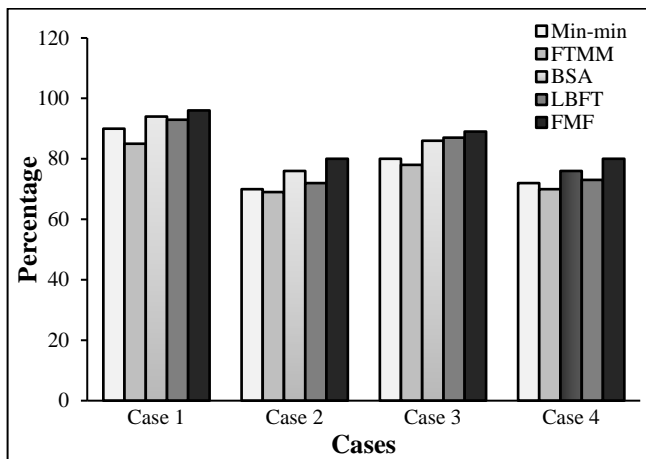


Fig.5. Average Resource utilization comparison of different algorithms with FMF

**6. CONCLUSION**

In order to consider user satisfaction, Fault tolerance, Load balancing and Resource allocation our proposed LSA algorithm implemented. This provide efficiency with resource allocation, Job scheduling with fault tolerance and it also dynamic sharing of the resource configuration has produced the resource contention in the grid computing, it has been resolved through job scheduling through Timely acquiring resource status information. Hence the fault tolerance of the resource in job scheduling has carried out with static multidimensional condition which updates the global

manager of the grid resource. But as a whole, it has a prominent improvement in makespan which proves improved system performance.

**REFERENCES**

- [1] V. Indhumathi, "Improved Fault Tolerant in Workload Execution through Quality Particle Swarm Optimization for Grid Environment", *Proceedings of International IEEE Conference on Computing for Sustainable Global Development*, pp. 5040-5045, 2016.
- [2] V. Indhumathi and G.M. Nasira, "Resource Monitoring and Prediction with Fault Tolerance in Grid Environment through Meta heuristics", *International Journal of Applied Engineering Research*, Vol. 10, No. 24, pp. 43993-44000, 2015.
- [3] G.M. Nasira and V. Indhumathi, "Fault Tolerance within Grid Environment using Platform LSF", *Proceedings of National Conference on Soft Computing*, 2012.
- [4] G.M. Nasira and V. Indhumathi, "Fault Tolerance within Grid Environment using Platform LSF", *Proceedings of National Conference on Advances in Computer Applications*, 2012.
- [5] Elvin Sindrilaru, Alexandru Costan and Valentin Cristea, "Fault Tolerance and Recovery in Grid Workflow Management", *Proceedings of International Conference on Complex, Intelligent and Software Intensive Systems*, pp. 475-480, 2010.
- [6] Yuzhong Sun and Zhiwei Xu, "Grid Replication coherence Protocol", *Proceedings of 18<sup>th</sup> International Symposium on Parallel and Distributed Processing*, pp. 232-239, 2004.
- [7] Wei Luo, Xiao Qin, Xian-Chun Tan, Ke Qin and Adam Manzanare, "Exploiting Redundancies to Enhance Schedulability in Fault-Tolerant and Real-Time Distributed Systems", *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, Vol. 39, No. 3, pp. 626-639, 2009.
- [8] Malarvizhi Nandagopal and Rhymend V Uthariaraj, "Hierarchical Status Information Exchange Scheduling and Load Balancing for Computational Grid Environments", *International Journal of Computer Science and Network Security*, Vol. 10, No. 2, pp. 177-185, 2011.
- [9] Jasma Balasangameshwara and Nedunchezian Raju, "A Hybrid Policy for Fault Tolerant Load Balancing in Grid Computing Environments", *Journal of Network and Computer Applications*, Vol. 3, No. 35, pp. 412-422, 2011.
- [10] Jia Yu and Rajkumar Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing", *Journal of Grid Computing*, Vol. 3, No. 3, pp. 171-200, 2005.
- [11] Aissatou Diasse and Foroski Kone, "Dynamic-Distributed Load Balancing for Highly-Performance and Responsiveness Distributed-GIS (D-GIS)", *Journal of Geographic Information System*, Vol. 3, pp. 128-139, 2011.
- [12] Renato Porfirio Ishii and Rodrigo Fernandes de Mello, "An Adaptive and Historical Approach to Optimize Data Access in Grid Computing Environments", *Infocomp Journal of Computer Science*, Vol. 10, No. 2, pp. 26-43, 2011.
- [13] Nhan Nguyen Dang and Sang Boem Lim, "Combination of Replication and Scheduling in Data Grids", *International*

- Journal of Computer Science and Network Security*, Vol. 7, No. 3, pp. 304-308, 2007.
- [14] Suriya and Prashanth, "Review of Load Balancing in Cloud Computing", *International Journal of Computer Applications*, Vol. 10, No. 1, pp. 35-39, 2013.
- [15] Dimple Juneja and Atul Garg, "Collective Intelligence based Framework for Load Balancing of Web Servers", *International Journal of Advancements in Technology*, Vol. 3, No. 1, pp. 64-70, 2012.
- [16] Abhijit and S.S. Apte, "A comparative Performance Analysis of Load Balancing Algorithms in Distributed Systems using Qualitative Parameters", *International Journal of Recent Technology and Engineering*, Vol. 1, No. 3, pp. 175-179, 2012.
- [17] Ali M. Alakeel, "A Fuzzy Dynamic Load Balancing Algorithm for Homogeneous Distributed Systems", *World Academy of Science, Engineering and Technology*, Vol. 6, No. 1, pp. 7-10, 2012.
- [18] Sarpeet Singh and R.K. Bawa, "Proactive Fault Tolerance Algorithm for Job Scheduling in Computational Grid", *International Journal of Grid and Distributed Computing*, Vol. 9, No. 3, pp. 135-144, 2016.
- [19] Walaa Abd Elrouf, Adil Yousif and Mohammed bakri Bashir, "High Exploitation Genetic Algorithm for Job Scheduling on Grid Computing", *International Journal of Grid and Distributed Computing*, Vol. 9, No. 3, pp. 212-228, 2016.
- [20] Harkiran Kaur and Mandeep Kaur, "Comparative Study of Various Task Scheduling Algorithms in Grid Computing", *International Journal of Advanced Research in Computer and Communication Engineering*, Vol. 5, pp. 843-844, 2016.