Asian Journal of Computer Science and Technology
ISSN: 2249-0701 (P) Vol.1 No.1, 2012, pp.111-114

DOI: https://doi.org/10.51983/ajcst-2012.1.1.1696

RMS: A Flexible Approach for Fault Tolerant Mechanism in the Grid Environments

R.Sivapriya and M.Karthikeyan
Department of Computer Science and Engineering,
Mohamed Sathak Engineering College, Kilakarai, Ramanathapuram, Tamil Nadu, India
E-mail: sivapriyaraman@gmail.com

Abstract **- Grid computing is the major research area where the distributed resources are used. In Scheduling, the biggest challenge is to acquire optimum solution for the submitted jobs in the grid. For large subtask require time consuming computation, this paper introduces a new fault recovery mechanism into grid systems and an in depth study on grid service. We propose a new algorithm on considering these factors. In our proposed algorithm Recovery Mutual Scheduling, a catalog is used which will be responsive in accumulation of saving its state periodically. Consequently the overall throughput of a system is increased with the decentralized approach.**

Keywords **- Scheduling, decentralized approach, Process Repository, Process Manager, Status Collector, Process dispatcher, Catalog**

1. Introduction

With the emergency of recent technologies, the need of computing grows rapidly. In order to achieve better utilization of resources, all over the world, Grid is proposed. The Grid concept is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations [1].Grid Computing has been used in many scientific applications. A grid system manages a large number of heterogeneous resources.

In [2] the grid is also defined as "A type of parallel and distributed system that enables the sharing, selection and aggregation of geographically distributed autonomous and heterogeneous resources dynamically at runtime depending on their availability, capability, performance, cost, and user's quality of service requirements". It enables users to share a large number of distributed computing resources over a network. Nowadays, grid computing has been widely accepted, researched, and given attention to by researchers [3].

Generally, there are four major means foachieving reliable systems: fault prevention, fault tolerance, fault removal, and fault forecasting [4]. It is clearly impossible to avoid all the faults during its execution. Fault prevention, prevents the occurrence or introduction of faults.Fault tolerance, avoid service failures in the presence of faults. Fault removal,reduce the number and severity of faults. Fault forecasting, estimate the present number, the future incidence, and consequences of faults. From the above definition, Fault tolerance which aims at delivering a correct service even in the presence of faults becomes a preferred choice for reliable Grid services.

The rest of the paper is organized as follows: In Chapter 2 background work was discussed. Chapter 3 highlights previous research in the related area. Then, the computational algorithm of RMS method is depicted in Chapter 4 and proposed system architecture and their functionalities. Chapter 5 shows the Experimental result. Finally, the concluding remarks and future direction of this research work is described in Chapter 7.

II. Background Work

A. Fault

The term error and fault are the same. Two types of fault, they are permanent and temporary. Permanent Faults include connection disruption, hardware breakdown. When an error disappears shortly it is called temporary fault. Software error comes under temporary fault.

B. Fault Tolerance

Fault Tolerance is the ability of a system to perform its function correctly even in the presence of faults.

C. Fault Recovery

Fault recovery can be roll forward or roll back. Roll forward recovery takes the system state at that time and corrects it i.e.it is able to move forward. Roll back recovery takes the system back to earlier state i.e. correct state and moves forward from that place.

III. Related Work

Simplest failure recovery technique is the Retry fault tolerance technique. In this, Whenever failure occurs, it will not be encountered in subsequent retries [12].Replication based technique is one of the popular fault tolerance techniques [5]. The word replica means multiple copies. Among the set of replicas, request from client is forwarded to one of the replica. I results in redundancy. By the replication of data, failure of some nodes will not be affected. Some researchers adopt the use of replication as well as masking [6], [7]. It uses replicated servers to mask the failures. Another well known technique is checkpoint with rollback-recovery. Check pointing is more popular fault tolerant approach used in distributed systems

[13]. It stores the current state of computation periodically, which can be used when node failure occurs. When an error is detected, the process is roll backed to the last saved state [8]. To restrict the influence of failures to the other agents in the same communication group, the message logging [7] or the checkpointing coordination [8] can be employed.

Excessive checkpointing would result in performance degradation, while deficient checkpointing would still incur an expensive recovery overhead [9].If more number of checkpoints are used, it results in overhead. If less number of checkpoints are used, it results in re-computation overhead.

In paper [10] it describes, a flexible fault-tolerance mechanism implemented on Integrate grid middleware that allows the customization of several failure handling parameters and the combination of different failure handling techniques.In paper [11], Genetic algorithm (GA) for job scheduling on computational grids was proposed. 5 different parameters and 3 different operations are considered. Finally it would successfully execute the job by improving the reliability of the system.

Paper [14] based on replication techniques. Whenever an agent wants to move,is is replicated.For each group,there is a worker node and others act as observers of the worker node. Worker node is responsible for the execution of the agent, whereas others receive a copy of the agent.

In paper [15], limitations of paper[14] are overcomed. They use a different leader election protocol and commit the local transaction using a 3PC protocol. but it leads to the violation of exactly once property.

IV. Recovery Mutual Scheduling

The decisive factors of existing system like communication cost, Execution time and node failures occurred during execution are condensed in our proposed System. In this Recovery Mutual Scheduling (RMS) algorithm a catalog is used which will be responsive in accumulation of saving its state periodically? Consequently it is found that the overall throughput of a system is increased with the decentralized approach.

Agents' schedules user jobs on the best available resource by optimizing cost, time or cost/time. One of the significant Factors for grid environments is Job Scheduling .Subsequently allocating the job to the processor it starts its computation. While processing its computation may be stopped in the middle due to some reason. The reasons may be due to unavailability of resources insufficient memory etc. This proposed approach aims at recovering the process from faulty / erroneous situations.

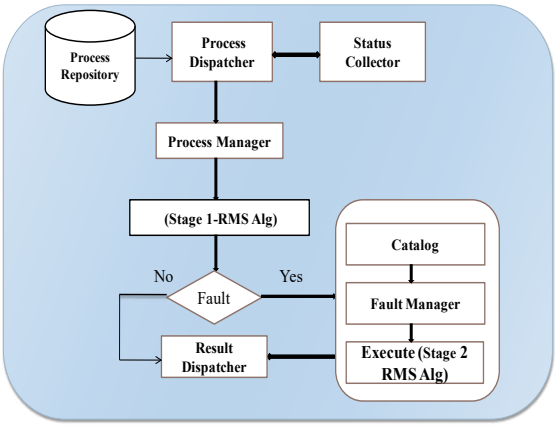

Fig.1 Data Flow Diagram

RMS algorithm works on two strategies. The RMS algorithm has been explained below.

RMS Algorithm (stage 1)

Step 1 If (ID=0)
Step 2 Accumulate tasks {Split Process into several tasks}
Step 3 Begin transaction
Step 4 Input (Parameters: Process Features)
Step 5 For (i=0 to njob) Execution of transaction
Initializing catalog registers each time
Checks for failure
Set ID=1
Activate the catalog
Transfer it to fault manager
Fault Manager ()
Return the ID if any failure
Step 6 Returns the result set
{jID, njob, tokens, jobfeature}

RMS Algorithm (stage 2)

Step 1 While (no task in pool)
Process List = Collect the Process
For(i=0 to ncid's)
Get the input Attributes
{jobid, cid,cstate,jobfeatures}

Procedure : Computetask()
Return the result to result DispatcherSet
status=0End

Procedure : ComputeTask()
Step 1 While(true)
Step 2 Set the input attributes
Step 3 Start the execution
Step 4 Find the Processors (Resources) to allocate its process
Step 5 Validate catalog at each quantum
Step 6 Return the result to result dispatcher
Step 7 If(fault occurred) Set status=1

Step 8 if (Status =1)
Step 9 Update the Fault manager;
Step 10 Dispatch it again to Catalog

When a task is submitted to a process manager, the execution of stage 1 of RMS algorithm begins. During its execution, its state is stored periodically in the catalog. On successful completion of task, the result is submitted to the Result Dispatcher. If any fault occurs, it contacts the fault manager for recovery consequently Stage 2 of RMS algorithm execution takes place. The execution of the task begins from the faulty state and proceeds further until the result is submitted to the result dispatcher.

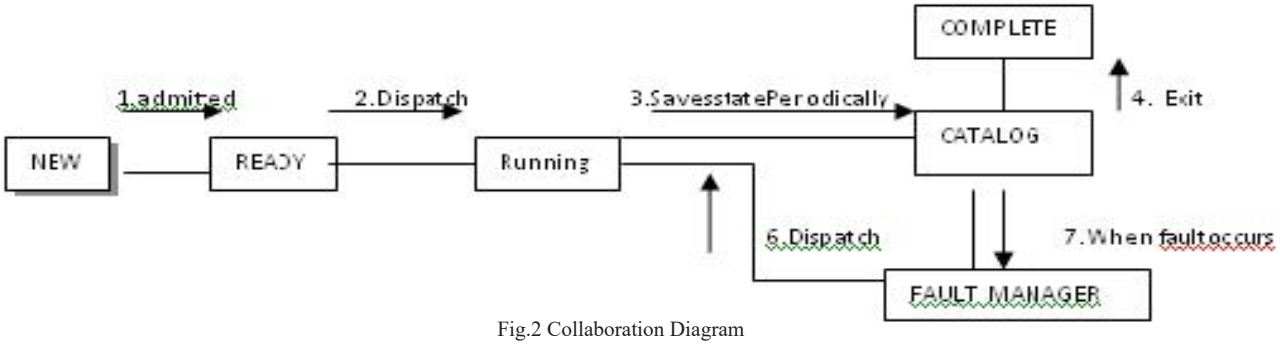

Fig.2 Collaboration Diagram

V. Experimental Studies

To study the performance of the algorithm, the proposed algorithm is compared with the existing algorithms multilevel hybrid scheduling algorithm and multilevel dual queue scheduling algorithm. Experimental results are carried out by simulation.

A. Performance Metrics

In our experiments, we measure the following metrics. Processing Time: the time it takes to complete a prescribed procedure

B. Results

In this experiment, we vary the Processor as 8, 16, 32 and 64.

Table I Processor Vs Processing Time (In Milliseconds)

Processer	Processing Time		
x	MHS	MDQS	RMS
8	13	26	10
16	15	24	9
32	14	20	7
64	8	16	3

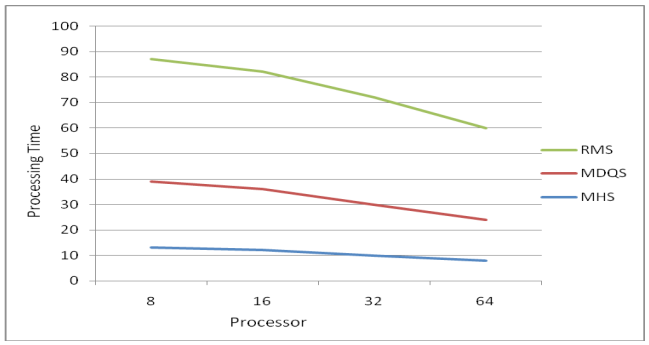

Fig.3 Comparison of MHSA,MDQSA and RMS

Figure 3 Depicts that No of Processors used by the three algorithms for processing is collectively combined with its time required to process .By means of Existing System algorithms for 8 Processor its processing time is 10 seconds whereas for MHS its 13 simultaneously for MDQS its 26. Consequently this graphical representation portrays that Time decreases as the number of processor increases. Moreover Time taken for processing is less when evaluated with previous techniques. In this experiment, we vary the Schedule Interval as 500, 1000, 1500, 2000 and 2500.

Table II Schedule Interval Vs Job Count(In Milliseconds)

Schedule Interval	Job Count		
x	MHS	MDQS	RMS
500	60	102	172
1000	75	116	182
1500	80	127	198
2000	88	136	212
`2500	99	154	232

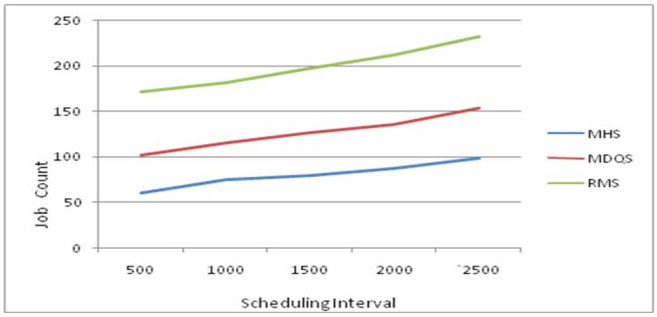

Fig.4 Comparison of MHSA,MDQSA and RMS

Figure 4 illustrates that Job count initialized by the proposed approach Increases as the Scheduling Interval Increases. In the previous approaches no of Job count is lesser when compared with the current One. For 2000 milliseconds as per

MHS algorithm it can schedule 88 jobs along with this for MDQS algorithm it can schedule 136 jobs. However as per RMS it can schedule 212 Jobs.In this experiment, we vary the Run Time as 100, 200, 300, 400, and 500.

The Pictorial representation between process Run Time and its count is shown in Figure 4.Its clear that as the Run Time and the process count are directly proportional to each other. Nevertheless applying this proposed approach the number of jobs taken to schedule is increasing when compare with the existing algorithms. Example within 400 ms RMS can execute 3212 number of jobs which is nearly more than 1000 jobs higher with the existing algorithms.It can be observed that the Proposed RMS algorithm has a better performance than the existing algorithms.

TABLE III RUN TIME VS PROCESS COUNT(IN MILLISECONDS)

Run Time	Process Count		
x	MHS	MDQS	RMS
100	1026	1658	2008
200	1352	1852	2412
300	1672	2012	2864
400	1842	2244	3212
500	2076	2656	3658

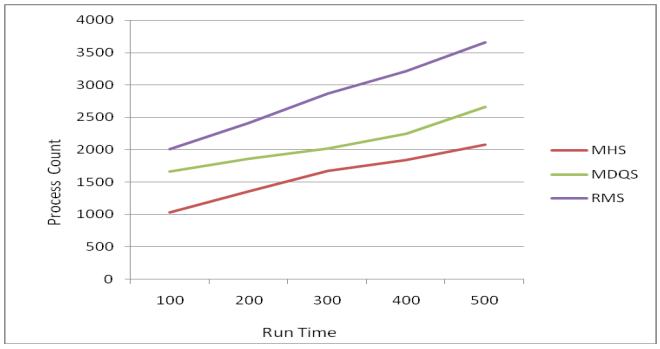

Fig.5 Comparison of MHSA,MDQSA and RMS

One of the key advantage of the proposed algorithm is Less communication cost, Processing time and node failures. The reason for this behavior is that the process does not roll back to previous state.

VI. CONCLUSION

This paper presented a flexible Recovery Mutual Scheduling RMS Algorithm. By using the proposed algorithm, less communication cost, execution time can be achieved. In Future we have planned to still improve the performance of RMS algorithm and implementing in dynamic grids dealing with graceful degradation.

REFERENCES

[1] Ian T. Foster, Carl Kesselman and Steven Tuecke, The anatomy of the grid enabling scalable Virtual Organisations, CoRR,cs. AR/0103025,2001

[2] M.Baker,R.Buyya and D.Laforenza," Grids andGrid Technologies for wide area Distributed Computing" in J*ournal of software-Practice& Experience* ,vol.32, no.15, pp.14371466,2002.

[3] I. Foster, "The Grid: A new infrastructure for 21st century science,"*Physics Today,* vol. 55, no. 2, pp42–47, 2002.

[4] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell and Carl Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing", *IEEE Trans, Dependable and Secure Computing,* vol. 1, no. 1, Jan-Mar 2004.

[5] M. Strasser and K. Rothermel. "System Mechanism for Partial Rollback of Mobile Agent Execution", *Proc. 20th Int'l Conf. on Distributed Computing Syst.*, pp. 20-28, 2000.

[6] S. Pleisch and A. Schiper. "FATOMAS - A Fault-Tolerant Mobile Agent System Based on teh Agent-Dependent Approach", *Proc. Int'l Conf. on Dependable Syst. and Networks,* pp. 215-224, 2001.

[7] S. Pleisch and A. Schiper. "Modeling fault-tolerant mobile agent execution as a sequence of agreement problems", *Proc. of 19th IEEE Symposium on Reliable Distributed Systems* (SRDS'00), Nuremberg, Germany, pp. 11-20, 2000

[8] E. Gendelman, L.F. Bic, and M.B. Dillencourt. "An Application-Transparent, Platform-Independent Approach to Rollback-recovery for Mobile Agent Systems", *Proc. 20th Int'l Conf. on Distributed Computing Syst.,* 2000.

[9] Y. Ling, J. Mi, and X. Lin. "A variational calculus approach to optimal checkpoint placement", *IEEE Transactions on Computers,* Vol. 59, No. 7, pp. 699-708, 2001.

[10] Stanley Araujo de Sousa, Francisco Jos´e da Silva e Silva," A Flexible Fault-Tolerance Mechanism for the Integrade Grid Middleware".

[11] Javier carretero, fatos xhafa, ajithabraham, Genetic Algorithm Based Schudulers for Grid Computing System.

[12] Soon Hwang and Carl Kesselman "A Flexible Framework for Fault Tolerance in the Grid", *Journal of Grid Computing* 1: 251–272, 2003.

[13] Pankaj Jalote, "Fault Tolerance in DistributedSystems", ISBN: 0-13-301367-7, 1994.

[14] K. Rothermel and M. Strasser. "A fault-tolerant protocol for providing the exactly-once property of mobile agents", *Proc. of the 17th IEEE Symposium on Reliable Distributed Systems* (SRDS'98), Dana Point, CA, pp. 138-147, 1998.

[15] F.M. Silva and R. Popescu-Zeletin. "An approach for providing mobile agent fault tolerance", *Proc. of the 2nd Int. Workshop on Mobile Agents* (MA'98), K. Rothermel and F. Hohl, Eds. LNCS 1477. Springer-Verlag, New Youk, 14-25, 1998.