Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 5, May 2014, pg.547 – 551

RESEARCH ARTICLE

Fault Tolerance Testing for Crash and Omission Transient Failure during Resource Scheduling of Grid Computing

Inderpreet Kaur[1], Sarpreet Singh[2]

[1]Department of Computer Science Engg., Sri Guru Granth Sahib World University, Fatehgarh Sahib, Punjab, India

[2]Department of Computer Science Engg., Sri Guru Granth Sahib World University, Fatehgarh Sahib, Punjab, India

Email id: inderpreetrao@gmail.com, ersarpreetvirk@gmail.com

Abstract:- In computational Grid, fault tolerance is an imperative issue to be considered during job scheduling. Due to the widespread use of resources, systems are highly prone to errors and failures. Hence fault tolerance plays a key role in grid to avoid the problem of unreliability. The two main techniques for implementing fault tolerance in grid environment are check pointing and replication. Grid Computing involves a network of computers that are utilized together to gain large supercomputing type computing resources. Scheduling the task to the appropriate resource is a vital requirement in computational Grid. This paper presents an overview of Resource Management; its basic function and structure, fault tolerance techniques. The proposed method is to improve one of the Fault Tolerance Algorithm that is the fittest resource scheduling algorithm, by scheduling the job in coordination with job replication when crash occurs.

Keywords: - Resource Management, Fault tolerance, Task Replication, Job Scheduling

I. INTRODUCTION

The term resource management in grid computing can be defined as those operations that control the way that grid resources and services are made available for use by entities like users, applications and services to ensure efficient utilization of computer resources and for optimization performance of specific tasks [1].

Resource management is a complex task involving security, fault tolerance along with scheduling. It is the manner in which resources are allocation, assigned, authenticated, authorized, assured, accounted, and audited. Resources include traditional resources like compute cycles, network bandwidth, space or a storage system and also services like data transfer, simulation etc.

A. Resource Management System (RMS)

The basic structure and functions of the RMS in the grid have been introduced and classified the existing RMSs into different categories based on their control property and applications [2],[3],[4]. Most kinds of RMSs have a common service process as in Fig.1. At first, the grid jobs/programs submit their requests for some resources to the RMS. The RMS adds these requests into the request queue [2]. Then, the RMS schedules and allocates those requests to different RM servers for matchmaking. In the matchmaking, the RM server discovers the requested resource [5] and then builds the connection between the request and the resource. Finally, the grid jobs/programs can access their requested resources through the constructed links.

The serving process of the RMS can be generally represented in Fig. 1. Grid jobs arrive at the RMS and request their needed resources. Translating the jobs is the first step that abstracts resource requests out of the jobs and put those translated requests into the request queue. The RMS then allocates those requests to different RM servers for the matchmaking in turn. Finally, the jobs can access those resources through the network.

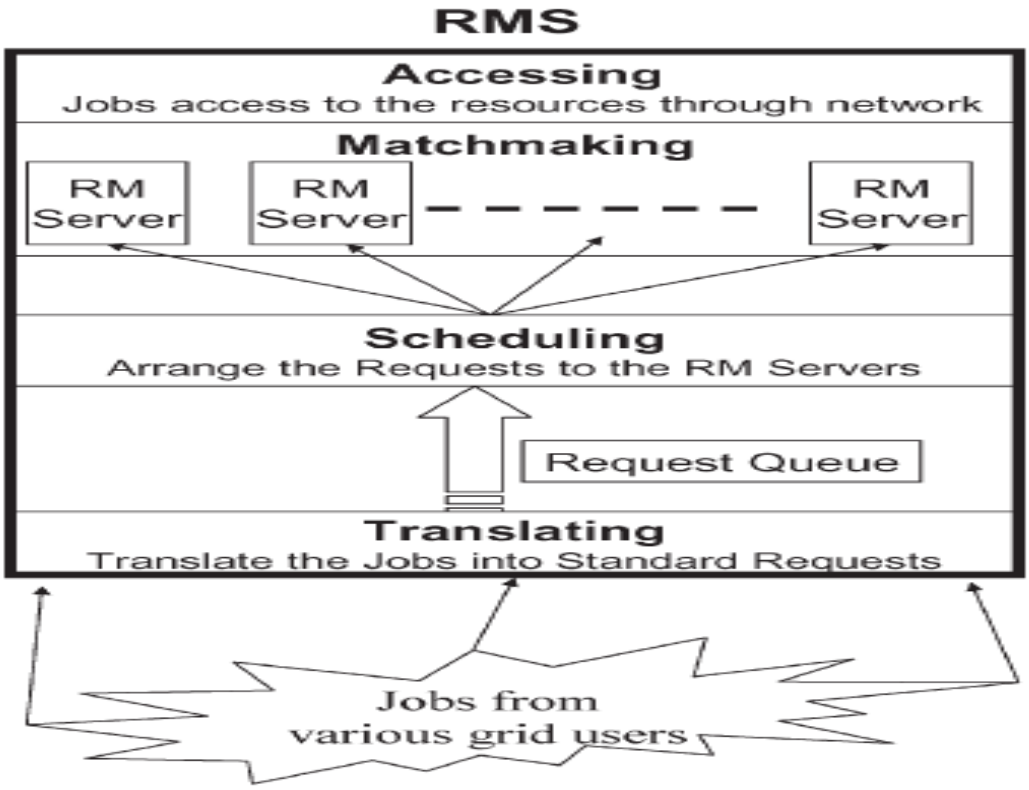

Fig.1. Serving process of the RMS

B. Fault Tolerance Transient Failure

In the grid environment the resources are heterogeneous and highly distributed. Hence they are prone to failures. Any scheduling algorithm will be more effective if fault tolerance is taken into account.

In the grid computing environment, when a resource is over exploited that is, if a resource is allocated to many Gridlets, it depletes the power of the resources leading towards its failure. The execution time also increases for the job which is assigned to these resources. Nodes can also crash and omission failures occur.

To solve this problem, the less reliable resources are identified as critical resource on the basis of failure rate of the resource. For all the resources which are less reliable, when a gridlet is allocated to it, there are greater probability of failure which can cause severe problems for the users of the Grid environment.

For this, Gridlets which are assigned to those critical resources are replicated to another resource in the same level. This duplicated gridlet is now again kept on the Grid List for execution. So the chances of the successful execution of the submitted Gridlet will increase. The resource which fails for a particular time is not assigned to any other job during that period. The failed jobs are again added to the Gridlet List and are ready for the scheduling again.

These jobs are allocated to the same level resource if any resource is present in same level or they are allocated with the resource of next level. This concept of replicating of the job will increase the reliability of the simulator. And also reduce the overhead of the resources. The fault tolerance is the imperative issue of this work. If node haven't any enough space to take any job, it hand over its job to other node on the same level. It shows availability of crash failure and omission failure of node.

II. RELATED SURVEY

Fault Tolerance plays a key role in grid to avoid the problem of unreliability. There are following techniques for implementing fault tolerance in grid environment.

A. Replication technique to improve the fault tolerance of the fittest resource scheduling algorithm [6]. The fittest resource scheduling algorithm searches for the appropriate resource based on the job requirements, in contrary to the general scheduling algorithms where jobs are scheduled to the resources with best performance factor.

B. The check pointing is one of the most popular technique to provide fault-tolerance on unreliable systems. It is a record of the snapshot of the entire system state in order to restart the application after the occurrence of some failure. The checkpoint can be stored on temporary as well as stable storage [7]. However, the efficiency of the mechanism is strongly dependent on the length of the check pointing interval. Frequent check pointing may enhance the overhead, while lazy check pointing may lead to loss of significant computation. Hence, the decision about the size of the check pointing interval and the check pointing technique is a complicated task and should be based upon the knowledge about the application as well as the system.

Therefore, various types of check pointing optimization have been considered by the researchers, e.g., (i) Full check pointing or Incremental check pointing (ii) Unconditional periodic check pointing or Optimal (Dynamic) check pointing (iii) Synchronous (Coordinated) or asynchronous (Uncoordinated) check pointing, and (iv) Kernel, Application or User level check pointing.

C. In pro-active mechanisms, the failure consideration for the grid is made before the scheduling of a job, and dispatched with hopes that the job does not fail. Whereas, post-active mechanisms handles the job failures after it has occurred. However, in the dynamic systems only post-active mechanism is relevant [8].

D. In order to detect occurrence of fault in any grid resource two approaches can be used: the push or the pull model. In the push model, grid components periodically send heartbeat messages to a failure detector, announcing that they are alive. In the absence of any such message from any grid component, the fault detector recognizes that failure has occurred at that grid component. It then implements appropriate measures dictated by the predefined fault tolerance mechanism. In contrast, in the pull model the failure detector sends live-ness requests ("Are you alive?" messages) periodically to grid components. [9].

E. Hwang et al. [10] presented a failure detection service (FDS) and a flexible failure handling framework (Grid-WFS) as a fault tolerance mechanism on the grid. The FDS enables the detection of both task crashes and user defined exceptions. Like any middleware, a grid middleware is also responsible to hide, from the application developer, the technical details related to different syntax and access methods and to provide a consistent and homogeneous access to resources managed locally.

F. Mohammad et al. [11] use agents to inject proactive fault tolerance in grids .Here autonomous, light-weight, intelligent agents monitor with individual faults. Agents maintain log of various information related to hardware conditions, memory utilization, resource constraints, network status and component failure. Based on this information and critical states, agent can enhance the reliability and efficiency of grid services.

G. F. Berman, H. Casanova, and A. Chien presented a rescheduling approach using stop and restart. The motivation for rescheduling is similar to the one, rescheduling a job to a better resource. However, a similar strategy can be adopted for fault tolerance. In[12] when a running application is signaled to migrate, it checkpoints user-specified data. For fault-tolerance, instead of user-directed check pointing such as the one in [12], an automatic check pointing approach has to be followed. In automatic check pointing, the state of the program is saved periodically to a persistent storage. When the machine running the program crashes, the program can be rescheduled to run on a different machine, continuing from the last check pointed state.

H. Distributed Fault-Tolerant Scheduling (DFTS) [13] policy is based on the job replication mechanism. It does not reschedule the job when a fault occurs. Instead it schedules more than one copy of the job (called replicas) to a different set of resources. The job is considered complete if one of the set of resources successfully executed the complete job. The underlying assumption for job replication is that in a grid many resources may be underutilized.

I. D.Saha et.al.,[14] describes the FPLT algorithm in which the scheduler sorts the node and task information by each node's CPU speed and task's workload in order to reduce complexity for searching the fastest node and the largest task all the time. Then the scheduler assigns the largest task of the waiting tasks to the available fastest node. FPLTF will reduce to the same as Work queue when the tasks arrive one by one.

J. Wang et.al [15] in their MFTF algorithm declares the fitness between the task and the node based of the expected execution time and execution time of the node. The node which has much less difference is expected to be the fittest node and the task is allocated to it. The fitness definition seems to be arbitrary.

K. Bajaj and Aggarwal [16] proposed an algorithm TANH (Task duplication-based scheduling Algorithm for Network of Heterogeneous systems) in which a new parameter is introduced for each task: the favorite processor (fp), which can complete the task earliest. Other parameters of a task are computed based on the value of fp. In the clustering step, the initial task of a cluster is assigned to its first fp, and if the first fp has already been assigned, then to the second and so on. Duplication based algorithms are very useful in Grid environments. The computational Grid usually has abundant computational resources (recall that the number of resource is unbounded in some duplication algorithms), but high communication cost. This will make task duplication very cost effective.

III. PROPOSED WORK

As the most fitting resource algorithm [6] the closeness factor which describes the appropriateness of the resource with the job requirements. The scheduling algorithm schedules the job to the appropriate resource rather the best performing resource. In this a new parameter called Reliability index is proposed to improve the fault tolerance of the most fitting resource scheduling algorithm.

When work load increases then the chances for the node failure increases. So once the node fails the jobs allocated to that node fails. If a node fails then we can't be sure that the Gridlet submitted to that particular node will execute successfully. This problem will lead to the wastage of time as well as the resource and due to the average waiting time for the Gridlet.

To get rid of this problem the reliability of the node is measured at the time of scheduling. If the most fitting node has less reliability level, the task is replicated to node in the same level.

The tasks that are handled by the crashed node will be replicated and new Gridlets ensures that the Job will be executed successfully. This will reduce the risk of job failure also.

In any case, if any nodes break down occurs, then we will analysis node for omission and crash transient factors. Every node will be having job assigned and shall be working independently in a connected grid network. The fault tolerance will behave as a non disturbing element. These factors will not affect the whole computational grid environment of the system.

IV. CONCLUSION

In the grid environment the resources are heterogeneous and highly distributed. Hence they are prone to failures. Any scheduling algorithm will be more effective if fault tolerance is taken into account. The fault tolerant most fitting resource scheduling was implemented in a java based grid simulator and the results are evaluated and concluded that with fault tolerance feature added to the scheduling algorithm. At the end of proposed work our scope would be to test the algorithm for crash and omission failure.

REFERENCES

[1] Humphrey, M., Thompson, M., Jackson, K. (March) 2005.Security for grids. Page 644-652 Proc. Of IEEE
[2] M. Livny and R. Raman, "High-throughput resource management," in *The Grid: Blueprint for a New Computing Infrastructure*. San Francisco, CA: Morgan Kaufmann, 1998, pp. 311–338.
[3] J. Nabrzyski, J. M. Schopf, and J. Weglarz, *Grid Resource Management*. New York: Kluwer, 2003.
[4] K. Krauter, R. Buyya, and M. Maheswaran, "A taxonomy and survey of grid resource management systems for distributed computing," *Softw.—Pract. Exp.*, vol. 32, no. 2, pp. 135–164, Feb. 2002.
[5] Q. Ding, G. L. Chen, and J. Gu, "A unified resource mapping strategy in computational grid environments," *J. Softw.*, vol. 13, no. 7, pp. 1303–1308, 2002.
[6] Ruay-Shiung Chang, Chun-Fu Lin, Jen-Jom Chen,"Selecting the most fitting resource for task execution" *Future Generation Computer Systems*, Volume 27, Issue 2, February 2011, Pages 227-231
[7] Oliner, A.J., Sahoo, R.K., Moreira, J.E., Gupta, M.: "Performance Implications of Periodic Check pointing on Large-Scale Cluster Systems", In Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, Washington, 2005.
[8] R. Medeiros, W. Cirne, F. Brasileiro, J. Sauve, "Faults in grids: why are they so bad and what can be done about it?" In proceedings of the 4th international workshop, November 2003, pp 18–24.
[9] Y. Li, Z. Lan, "Exploit failure prediction for adaptive fault-tolerance in cluster".In: Proceedings of the sixth IEEE International symposium on cluster computing and the grid, Vol 1, May 2006, pp.531-538.
[10] S. Hwang and C. Kesselman, "A Generic Failure Detection Service for the Grid", Technical Report ISI-TR-568, USC Information Sciences Institute, 2003.
[11] Mohammad Tanvir Huda, Heinz W. Schmidt, Ian D. Peake, "An Agent Oriented Proactive Fault-Tolerant Framework for Grid Computing", In Proceedings of the First International Conference on e-Science and Grid Computing, 2005, pp.304-311.
[12] F. Berman, H. Casanova, and A. Chien. New grid scheduling and rescheduling methods in the GrADS project. *International Journal of Parallel Programming*, 33(2–3):209–229, June 2005
[13] Jemal H. Abawajy. Fault-tolerant scheduling policy for grid computing systems. In *Proceedings of 18th International Parallel and Distributed Processing Symposium*, volume 14, page 238, April 2004.
[14] D. Saha, D. Menasce, S. Porto, et al., "Static and dynamic processor scheduling disciplines in heterogeneous parallel architectures", *Journal of Parallel and Distributed Computing* 28 (1)(1995) 1–18.
[15] S. Wang, I. Hsu, Z. Huang , "Dynamic scheduling method for computational grid Environments" , *in: Proceedings of the International Conference on Parallel and Distributed Systems*, July 2005, pp. 22–28.
[16] S. Ranaweera and D. P. Agrawal, "A Task Duplication Based Scheduling Algorithm for Heterogeneous Systems", *in Proc. of 14th International Parallel and Distributed Processing Symposium (IPDPS'00*), pp. 445-450, Cancun, Mexico, May 2000.

