A Novel Technique for Task Re-Allocation in Distributed Computing System

Madhav Gupta
Mtech student (ECE departent)
Chandigarh engineering college
Mohali, India
Madhavgupta890@gmail.com

Dr. Ruchi Singla
Professor and Head, Department of ECE
Chandigarh engineering college
Mohali, India
cecm.ece@gmail.com

Abstract: A distributed computing is software system in which components are located on different attached computers can communicate and organize their actions by transferring messages. A task applied on the distributed system must be reliable and feasible. The distributed system for instance grid networks, robotics, air traffic control systems, etc. exceedingly depends on time. If not detected accurately and recovered at the proper time, a single error in real time distributed system can cause a whole system failure. Fault-tolerance is the key method which is mostly used to provide continuous reliability in these systems. There are some challenges in distributed computing system such as resource sharing, transparency, dependability, Complex mappings, concurrency, Fault tolerance etc. In this paper, we focus on fault tolerance which is responsible for the degradation of the system. A novel technique is proposed based upon reliability to overcome fault tolerance problem and re-allocate the task.

Keywords*: DCS, reliability, master node time, Failure rate, execution time.*

*****

I. Introduction

Computing is an activity which is used for creating, designing and managing applications. Computing includes all the software and hardware schemes, structures and emergent processes and manages them properly. Parallel and distributed are the types of computing. In parallel computing, memory is shared between all the processors to exchange information. Bit-level, instruction level and data level are its type.

A distributed computing is software system in which components are located on different attached computers communicate and organize their actions by transferring messages [7]. BOINC is the example of DCS. It is an outline in which large tasks are divided into many little tasks that are further distributed to many workstations. Then, these small results are reassembled into a larger solution. Cloud Computing is one of the types of distributed computing. Cloud computing is a branch of computing that comes under the category of sharing of data instead of local servers. This policy of distributed computing is completed through polling of all computer resources together. It permits user and its client to access their personal files at any computer with the help of internet. Grid computing is another type or category of distributed computing. Grid computing is a computing environment with high performance to solve larger scale computational problems. There are number of grids that are present in grid computing which combine together to form a cluster that helps to reach at the common goal [5]. The grid clusters are not come under the single administration. The users of grid admit all the grids as a computing source which executes the job efficiently at any resource.

1.1 Challenges of Distributed Computing System: There are many challenges in distributed computing system as follows:

1. Resource Discovery and Selection: The dynamic environment of the wireless mobile makes necessary use of sophisticated mechanisms for resource discovery and selection. The list of the authorized machines and resources that are available in the mobile infrastructure are continuously updated. Some of the required parameters are resource accessibility, system workload, network performance, etc. A financial criterion of the resources used should be under consideration for the proper resource selection [5].

2. Information Security: Due to their wireless nature, devices in the mobile can communicate and pass information over standard radio frequencies that can be easily tapped. A number of encryption standards such as WEP have been devised to ensure data security and integrity [4].

3. Job Replication, Migration and Monitoring: In such context, job migration and re-scheduling as well as job replication and co-scheduling are both efficient ways to guarantee the completion of the jobs according to the restrictions set by the users. Job monitoring is difficult in dynamic environments. It is responsible for detecting alert situations that could initiate a migration and alternatively identify if a job has been completed so as to suspend/stop/cancel the execution of the other replicas.

II. Review of Literature

This paper [1] presented an efficient solution to the dynamic allocation problem. Starting with the definition of the phase of a modular program, a model based on dynamic programming approach is suggested. Earlier the researchers advised that the dynamic allocation strategy is the best allocation technique as it facilitates the user to take decision for allocating during run time. The suggested algorithm is implemented on the several sets of input data and it is recorded that algorithm is workable in all the case. Here, they have considered the phases and each phase has the tasks that are to be processed by the processors. In each phase only one task will be executed on these processors. During the next phase an executing task may remain on the same processor for execution or may shift to another processor in case of shifting the task to another processor, it added the reallocation cost. The impact of inter task communication cost is to be considered. Thus, phase wise optimal costs are obtained. In this model, there are five phases and each phase has the equal number of tasks. Optimal allocation has been obtained along with phase wise optimal costs.

In this paper [2], they addressed the distributed tracking control problem for multi-agent system with heterogeneous uncertainties and a leader whose control input might be nonzero and not available to any follower. Based on the relative state of neighboring agents, both distributed continuous static and adaptive controllers have been designed to guarantee the uniform ultimate boundedness of the tracking error for each follower. A sufficient condition for the existence of these distributed controllers is that each agent is stabilizable.

In this paper [3], they represented Mobile Agent technology that promises to be a powerful agent to know where the agent is and what is it mechanism to improve the flexibility and doing. Mobile agent systems must also provide a customizability of applications with its ability to provide an additional feature for the security of the host from a network. But none of the present mobile agent has malicious agents. The architecture proposed in this paper prototype systems satisfy all the requirements to address the above issues can be used to extend to provide a secure and reliable architecture, suitable for features of the existing systems.

In this paper [4], they presented a service for fault tolerance which includes an algorithm of modeling in group in the Ad hoc networks for then applying the tolerance to the faults by replication which is based primarily on the prediction. The definite algorithm forms the groups on the basis of number of neighbors and energy level of nodes. The network after clustering has a hierarchical structure of two levels with a leader for each group and a super leader (generic) for all networks. For the futures works: we wish to implement our service of fault tolerance in a simulator of network Ad hoc such as NS2 or GloMo Sim and improve our algorithm used in sub service of clustering by taking into account of intention of nodes and to integrate into our simulator various protocols of routings for the Ad hoc networks.

III. Load Balancing in Distributed System

Load balancing is traditional and conventional method so it does not suit with grid computing and distributed system environment [5]. Because these two environments are completely different, they have following problems:

1. Heterogeneity: Heterogeneity is present in both computational and network resources. First of all network used in grids are having different bandwidth and communication protocols. Secondly computational resources are heterogeneous. Software which are used by them like files system, operating systems are also heterogeneous [6].

2. Autonomy: Because there are multiple administrative domains that share mobile resources, a site is viewed as an autonomous computational entity. It generally has its individual scheduling policy which complicates the task allocation trouble. A single overall performance goal is not viable for a mobile system since each location has its own routine goal and scheduling decision is made independently of other sites according to its own performances.

3. Adaptability and scalability: A grid grows exponentially from few resources to millions. As the size of the grid increases, it creates the problem of network degradation. First of all it cannot provide guarantee of the bandwidth during the execution of the shared resources. It happens mostly in Wide Area Network like internet. Secondly, both the computational and distributed environment shows the dynamic behavior. Some resources or grid join the network simultaneously at the same time when resources become unavailable. They can extract maximum resources information from the resources [10].

4. Resource Selection and Computation: During the tenure of conventional and traditional methods both input and output are usually execute from the same site. Determination of the destination of the input and output is necessary before the submission of an application [11]. But in grid computing execution site is selected by the grid scheduler according to the some criteria of the performance of the resources. Bandwidth resources are limited and its cost of communication cannot be neglected. These challenges pose significant obstacles on the problem of designing an efficient and effective load balancing system for Grid environments. Some problems resulting from the above are not solved successfully yet and still open research issues. Thus it is very difficult to define a load balancing system which can integrate all these factors [9].

IV. Proposed Methodology

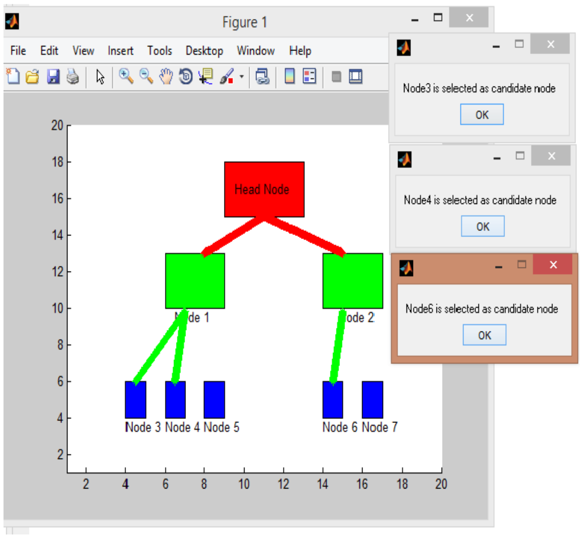
Distributed system plays an important role in achieving good performance and high system utilization. The goal of a job scheduling system is to efficiently manage the distributed computing power of workstations, servers, and supercomputers in order to maximize job throughput and system utilization. With the dramatic increase of the scales of today's distributed systems, it is urgent to develop efficient job schedulers. The number of users of distributed systems and networks considerably increase with the increasing complexity of their services and policies, system administrators attempt to ensure high quality of services. Each user requires maximum utilization of system resources. To achieve this goal, correct, real-time and efficient management and monitoring mechanisms are essential for the systems. But, as the infrastructure of the system rapidly scale up, a huge amount of monitoring information is produced by a larger number of managed nodes and resources so that the complexity of network monitoring function becomes extremely high. Thus, mobile agent-based monitoring mechanisms have actively been developed to monitor these large scale and dynamic distributed networked systems adaptively and efficiently. The proposed algorithm is assign task to other nodes only when master node moves from its original position. The major problem in this architecture is task scheduling, if one slave node get failed the task allocated by master node will not get completed and fault occurred. In this methodology, we will work on technique which helps to reduce fault tolerance of the system and increase performance of the system.

In this work, new formula of reliability is added which will calculate reliability of each node which are responsible for the task execution. All available nodes have reliability value 1 in the starting phase of the project. The formula applied is based on the maximum failure rate and maximum execution time. Each node in the network has its own failure rate and execution time. On the basis of these two and number of tasks which are going to be executed, reliability value is calculated. The node which has maximum reliability value will execute the task on the faulty node whose position is changed. The formula for the calculation of reliability is given below:

1. AB=Maximum execution time+ Maximum Failure rate
2. BC=execution time of each node + failure rate of each node
3. DE=BC*number of task for execution
4. Reliability= AB-DE

V. Experimental Results

The whole scenario is implemented in MATLAB.



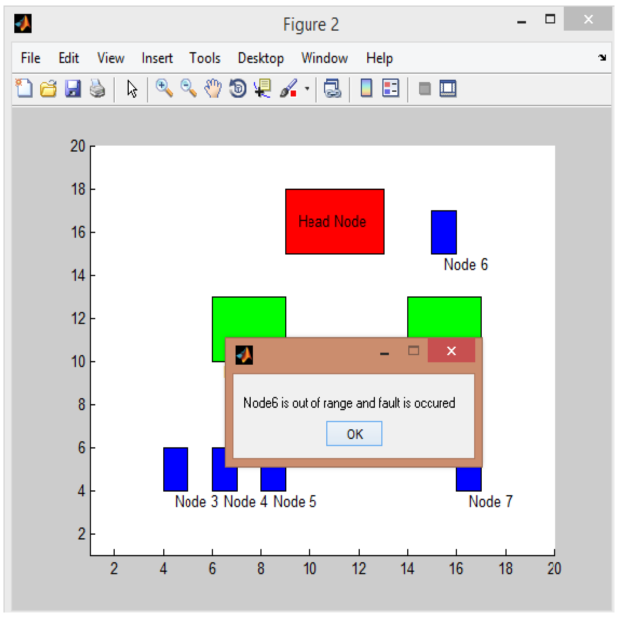
Node3 is selected as candidate node

Node4 is selected as candidate node

Node6 is selected as candidate node

Fig.1.1 Selection of Candidate Node.

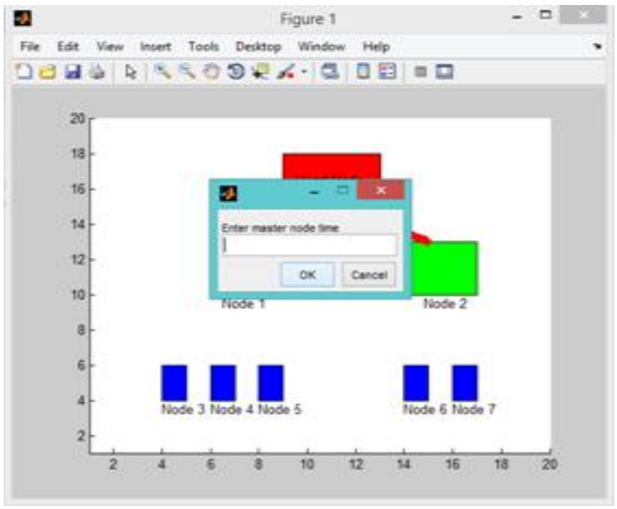
As illustrated in figure 1.1, the head node will assign task to its sub-nodes. On the basis of execution time and failure rate sub node allocates the task to node no 3, 4 & 6, as these nodes are selected as candidate nodes. Those nodes whose failure rate is less than the max failure rate and execution time is less than the max execution time are selected as candidate node.



Node6 is out of range and fault is occured

Fig.1.2: Fault Occurrence

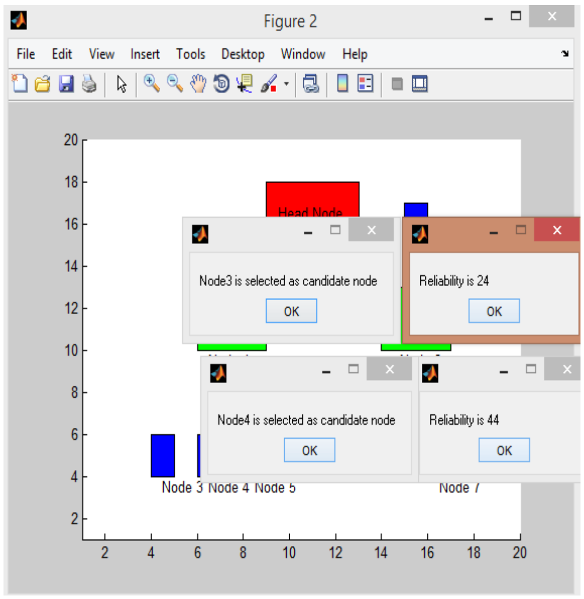
As illustrated in the figure 1.2, the candidate node 6 is out of range or changes its position due to mobility thus it is detected as a faulty node. The fault can also be occurred in the system when a node will not respond in its threshold time.



Enter master node time

Fig.1.3 Master Node time

As shown in figure 1.3, the sub-nodes are responsible to assign task to candidate nodes. The sub-node will ask for the number of tasks to be assigned to select candidate nodes. The candidate nodes when execute their assigned task will revert back to head node. In this figure, the user will enter the time taken by head node to merge result to the tasks.



Node3 is selected as candidate node

Reliability is 24

Node4 is selected as candidate node

Reliability is 44

Fig.1.4 Reliability of the node

As shown in figure 1.4, the reliability of every candidate node is calculated on the basis of applied formula. The reliability of node 3 is calculated as 24 and the reliability of node 4 is calculated as 44.

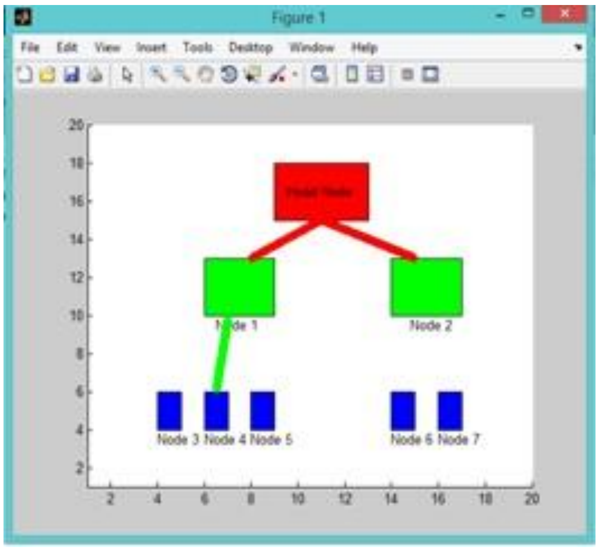
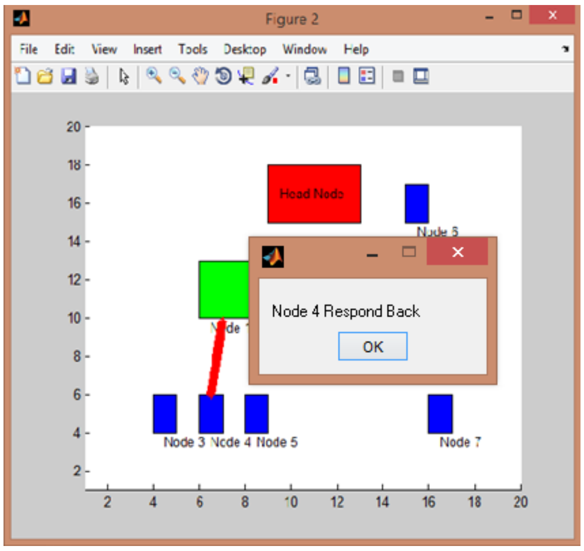


Fig.1.5 Task Re-allocation

As shown in figure 1.5, node 1 checks the reliability of both the candidate node 3 and 4. The node which has higher reliability will execute the task of the faulty node. The node 1 has then given the task of the faulty node (6) to node 4.



Node 4 Respond Back

Fig 1.6: Task reallocation.

As shown in the figure 1.6, the candidate node 4 which has higher reliability than the candidate node 3 will complete the task of the faulty node 6 and respond back to node1.

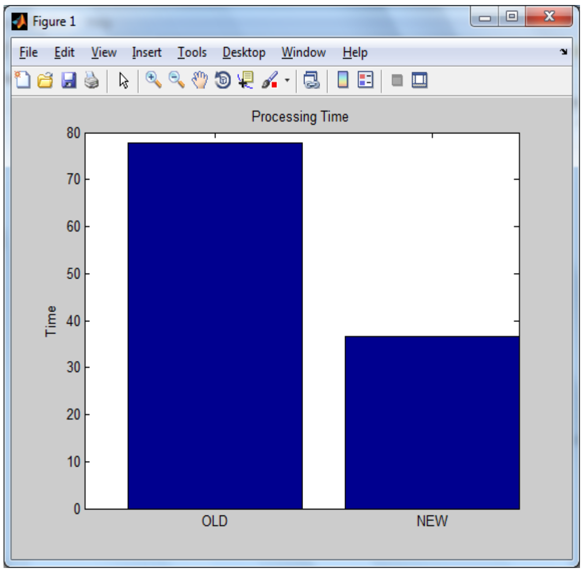


Fig 1.7: Time Graph

The figure 1.7 illustrates the graph between old versus new scenario showing the processing time of completion of task in case of faulty nodes. It is clear from the above graph that it consumes less time than the old methods.

VI. Conclusion and Future Scope

The load is equally divided among the mobile node to enhance the network efficiency and to reduce the task execution time. When the load is not equally divided among the mobile nodes, chance of error occurrence will be increased. The approach of fault tolerance is required to reduce the number of error rates in mobile distributed network. The task allocation among the mobile nodes is done with the use of task allocation model. In this paper, novel technique has been proposed which reduces the fault detection time in the network and reduces the resource consumption to execute the allocated tasks using reliability based technique. The proposed algorithm is based on the reliability of the node. The node having high reliability will be given the task of the faulty node. In future, enhancement in the proposed algorithm will be made to handle certain security attacks in mobile distributed system; these attacks are denial-of-service attacks which reduces the networks reliability and efficiency.

REFERENCES

[1] "Distributed Computing Principles, Algorithms, and Systems" by Ajay D. Kshemkalyani (University of Illinois at Chicago) and Mukesh Singhal (University of Kentucky, Lexington) © Cambridge University Press 2008, www.cambridge.org.

[2] Pradeep K. Sinha Distributed Operating Systems Concepts and Design, PHI Learning Private Limited, New Delhi- 110001, 2010.

[3] "Distributed Systems Concepts and Design" by George Coulouris, Jean Dollimore and Tim Kindberg, Addison-Wesley publication, Pearson Education 5th Edition 2011.

[4] Vinod Kumar Yadav, Mahendra Pratap Yadav and Dharmendra Kumar Yadav, "Reliable Task Allocation in Heterogeneous Distributed System with Random Node Failure", International Conference of Computing Science, vol 61, pp: 187-192, 2012

[5] Avizienis A., "Fault Tolerance Computing-An overview", IEEE Computer, Vol.4, pp. 5-8, Jan/Feb 1971.

[6] Rajwinder Singh and Mayank Dave, Senior Member, "Antecedence Graph Approach to Checkpointing for Fault Tolerance in Mobile Agent Systems", IEEE transactions on computers, vol. 62, no. 2, February 2013

[7] Sajjad Haider, Naveed Riaz Ansari and Muhammad Akbar, "Fault Tolerance in Distributed Paradigms", 2011 International Conference on Computer Communication and Management Proc .of CSIT vol.5 (2011) © (2011) IACSIT Press, Singapore.

[8] Zubair Khan, Ravinder Singh, and Jahangir Alam, "Task allocation using fuzzy inference in parallel and distributed system", Journal of Information and Operations Management, Volume 3, Issue 2, 322- 326, 2012

[9] Z. Li, X. Liu, W. Ren, and L. Xie, "Distributed tracking control for linear multiagent systems with a leader of bounded unknown input," IEEE Transactions on Automatic Control, vol. 58, no. 2, pp. 518-523,2013.

[10] Zibin Zheng and Michael R. Lyu, "Selecting an Optimal Fault Tolerance Strategy for Reliable Service-Oriented Systems with Local and Global Constraints" IEEE TRANSACTIONS ON COMPUTERS, VOL. 64, NO. 1, JANUARY 2015

[11] Jinho Ahn, "Lightweight Fault-tolerance Mechanism for Distributed Mobile Agent-based Monitoring" IEEE International Conference on Mobile Agent-based monitoring, 2009.

[12] Parmeet Kaur Jaggi and Awadhesh Kumar Singh, "Adaptive Checkpointing for Fault Tolerance in an Autonomous Mobile Computing Grid, IEEE, 2014.

[13] Zhe Wang, Naftaly and H. Minsky," Fault Tolerance in Heterogeneous Distributed Systems" 10th IEEE International Conference on Collaborative Computing: Networking, Applications and Work sharing (CollaborateCom 2014)

[14] Pritee Parwekar and Parmeet Kaur, "Fuzzy Rule based Checkpointing Arrangement for Fault Tolerance in Mobile Grids", IEEE2014.

[15] Rajwinder Singh, Mayank Dave, "Using Host Criticalities for Fault Tolerance in Mobile Agent Systems, 2nd IEEE International Conference on Parallel Distributed and Grid Computing, vol 2 pp:67-72, jun 2012

[16] Bahi, Jacques, Couturier, Raphael and Vernier, Flavien. Synchronous distributed load balancing on dynamic networks, Journal of Parallel and Distributed Computing, Elsevier Inc., Vol. 65, Issue 11, 1397 – 1405, 2005.

[17] Asma Insaf Djebbar and Ghalem Belalem, "Modeling by groups for faults tolerance based on multi agent systems", IEEE transactions on computers, vol.62, no. 6, February 2010.

[18] Zhongkui Li and Zhisheng Duan, "Distributed Tracking Control of Multi-Agent Systems with Heterogeneous Uncertainties", 10th IEEE International Conference on Control and Automation (ICCA) Hangzhou, China, June 12-14, 2013.