Weight based Task Assignment Model to Tolerate Faults in Heterogeneous Distributed Systems

Shubhinder Kaur
Student, Chandigarh University
Gharuan, India

Gurpreet Kaur
Assisatant Professor, Chandigarh University
Gharuan, India

ABSTRACT
Distributed frameworks play a critical part on accomplishing superior performance and better system utilization. The objective of a task allocation framework is to productively deal with the disseminated computing power of workstations, servers, and supercomputers keeping in mind the end goal to expand work throughput and system utilization. There are many issues of distributed computing system which are discussed in this paper in brief. This paper focuses on task assignment which in turn emphasizes on fault tolerance and recovery from fault with less processing time. The proposed algorithm assigns tasks to other nodes only when candidate node moves from its original position. The major area of concern in this architecture is task scheduling, if one slave node fails the task allocated by master node will not be completed and this situation is considered as fault. In this paper, we have exchanged views about a method which serves to lessen faults of the system and increase performance of the system.

Keywords
Distributed systems, Task allocation, job scheduling, and scalability.

1. INTRODUCTION
The distributed framework is an accumulation of autonomous PCs that seems to the clients as a solitary intelligible framework. A heterogeneous distributed computing system is that where random node can fail permanently. Since the DCS is heterogeneous, so its different nodes have diverse equipment and software characteristics. The diverse parts of the application likewise have different hardware and software prerequisites. Distributed computing take advantage of numerous PCs, each fulfilling a segment of total task, to accomplish a computational result more rapidly than with a solitary PC.A computer program that runs on distributed system is known distributed program. The process of writing such types of languages is called distributed programming [4]. Most importantly each computational element has local memory. The entity corresponds with every other entity with the assistance of message passing. Second the framework needs to endure itself from failures in individual PCs. The structure and the connections of framework may change amid the execution of distributed projects/programs. Every framework is aware about the information added by other framework. Resource or asset sharing is the capability to utilize any equipment, software or information anywhere in the framework. Assets in a distributed framework, dissimilar the centralized one, are physically encapsulated inside of one of the PCs and must be accessed from others by correspondence. Openness is uncertain with expansions and upgrades of dispersed frameworks. New components can be incorporated with present components so that the included usefulness gets accessible from the disseminated framework all in all [6].

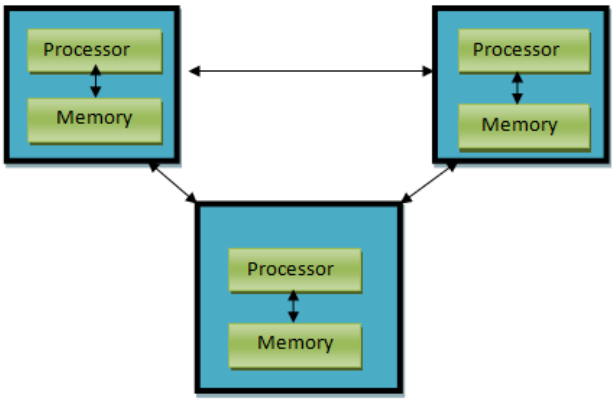

Fig.1 Distributed Computing Systems

1.1 Design Issues of Distributed System [16
A Distributed operating framework must be intended to give all the benefits of a distributed framework to its clients. There are some design issues which are as follows:

Flexibility: The distributed system should be flexible so that modifications and enhancement can be done easily by the users.

Scalability: System should be designed in this manner that it can easily cope up with increasing growth of the system. It should avoid central algorithms and central entities. It should perform most of the operation at the client work station. Distributed system operates at various levels ranging from intranet to internet. If the number of users and resources increases and the system still operates efficiently and effectively then it can be characterized as scalable. Cost of physical resources and performance management are the main challenges faced while designing a scalable distributed system.

Security: The different assets of a PC framework must be ensured against devastation and unauthorized access so that the clients can depend on the system. The security of information resources is one of the most important tasks as they are maintained and managed by distributed systems. There are three levels to maintain security of information resources—confidentiality, integrity and availability.

Fault Tolerance: Faults in distributed frameworks are fractional which implies a few components may fizzle and some still keep on doing their work. Therefore, handling of failure becomes difficult task. The system must be resistance to faults. In future if any fault occurs it doesn't degrade performance.

2. RELATED WORK

Paper [1] tended to the convey that the tracking control issue for multi-agent frameworks with heterogeneous instabilities and a pioneer whose control input may be nonzero and not accessible to supporters. Taking into account the relative conditions of neighboring agents, both disseminated continuous static and versatile controllers have been intended to ensure the uniform extreme boundness of the tracking error for every adherent. An adequate condition for the presence of these dispersed controllers is that every agent is stable.. In this paper [2] Mobile Agent technology promises to be a powerful of the agent, to know where the agent is and what is it mechanism to improve the flexibility and doing. Mobile agents systems must also provide customizability of applications with its ability to additional feature for the security for the agent from dynamically deploy application components across the malicious host and the security of the host from a network. But none of the present mobile agent is 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 [3] a service for issues resistance is presented which incorporates an algorithm for display in Ad hoc systems for applying the resilience to the deficiencies by replication which is constructed basically with respect to the forecast. On the premise of number of neighbors and vitality level of nodes the algorithm forms the group. The system subsequent to grouping has a progressive structure of two levels with a pioneer for every gathering and a super leader for all systems. The formed gatherings are open, dynamic, detached, unequivocal, and permitting point-to-point communication of group. The service of issues resilience connected is made out of four sub services to allow bunching, choice, replication by forecast and consistency improving it conceivable to deal with the system and which incorporates the functionalities vital for a superior accessibility of the data.

Paper [4] defines that the issue of the assignment of task in distributed processing framework is to apportion various tasks to diverse processors for execution. The paper manages the issue of task allotment in heterogeneous disseminated frameworks with the objective of maximizing the framework system reliability. They introduced a genetic algorithm to acquire the ideal solution for the issue. In the execution of the algorithm the parameters considered were the quantity of undertakings, the quantity of processors and task interaction density of various applications. The test results represent the viability of proposed algorithm over traditional algorithm. This paper [13] mentioned that the choice of load sharing strategy is one of the important part of distributed system design. The paper categorized load sharing algorithm into source initiative and server initiative and discussed various algorithms and evaluated their performance by using a metric called Q-factor. The paper [14] conferred that so as to minimize the acknowledgement time and enhance the performance of the disseminated frameworks the task assignment strategy concentrates on assigning tasks in efficient manner. The paper exhibited and talked about two classes of assignment task strategies: approaches which expect task size is known and approaches where task size is not known. Strategies which expect task size is known can further be classified as static or dynamic.

3. FAULT TOLERANCE IN DISTRIBUTED SYSTEMS

Distributing computing is a computational system in which software and hardware infrastructure provides consistence, dependable and inexpensive accesses to high end computations. An imperfect system because of a few reasons can bring about a few harms. A task which is chipping away at ongoing disseminated framework ought to be achievable, reliable and adaptable [6]. The ongoing disseminated frameworks like grid, robotics, nuclear airport regulation frameworks and so forth are very dependent on deadline. Any slip-up in real time distributed framework can bring about a framework into breakdown if not legitimately identified and recouped at time. Fault-tolerance is the important method which is often used to continue reliability in these systems. By applying extra hardware like processors, resource, communication links hardware fault tolerance can be achieved. In software fault tolerance tasks, to deal with faults messages are added into the system. Distributed computing is different from traditionally distributed system [7]. Fault Tolerance is important method in distributed computing because nodes are distributed geographically in this system under different geographically domains throughout the web. The most difficult task in distributed computing is design of fault tolerant task assignment model that meet all the reliability requirements [7]. The most commonly used techniques for fault-tolerance are replication and check pointing [15].

Replication: Replication is the method for keeping up diverse or various duplicates of data item. Replication includes redundancy in the framework such that failure of some node won't bring about failure of entire framework and consequently adaptation to non-critical failure is accomplished.

Checkpoint: The most ordinarily utilized fault tolerant technique is check pointing-restart. An application is restarted from an earlier check point or recuperation point after a shortcoming. Check pointing is utilized to abstain from losing all the helpful handling done before a flaw has happened.

A decent fault tolerant framework outline obliges a careful investigation of failures, reasons for those failures and framework reactions to those failures. Such learning should be approved out in aspect before the design start and have to remain part of the design process [9].Planning to keep away from failures is most important. A designer must examine the situation and decide the failures that must be tolerated to achieve the preferred level of dependability. To optimize fault tolerance, it is important to calculate approximately actual failure rate for each possible failure.

4. PROPOSED WORK

As distributed frameworks get complicated due to expanding client needs, observations and modifications are important to keep them fit and running. With the dramatic increment of the sizes of dispersed frameworks, it is the need of the hour to create efficient task assignment algorithms. The fast development in Internet clients and multiple services has highlighted the requirement for sensible tools that can help clients and applications in conveying the required quality of services. A node failure problem occurs due to mobility of the node. In the proposed scenario numbers of nodes are present. From the available nodes candidate node or nodes is/are selected which would take on extra work load in the event of failure. The master node sets a threshold value which includes parameters that are failure rate and maximum execution time

and master node time. Only those nodes are selected as candidate nodes which have failure rate and execution time less than the master node.

In the proposed algorithm, the head node will assign tasks to sub nodes and sub nodes are responsible for assigning task to the nodes. The sub node will select candidate node on the basis of parameters. These parameters are failure rate, execution time and master node time. Master node time is the resultant time to join the end users. Master node time is used for collaborating results from different nodes. Using these parameters we calculate weight of each node which contributes in selecting the best candidate node which can complete the task more efficiently.

Then we discuss the scenario of failure. During the mobility of node, the node which fails or moves from its location, if the node does not revert back to sub node in the decided time slot it will be considered as faulty node. Then the sub node will check for the most appropriate candidate node with maximum weight and the task of that node will be re-assigned to the selected candidate node which has highest weight.

The proposed algorithm consists of following three parts:

1. Reading the number of tasks and various parameters as input.
2. Selecting candidate node on the basis of parameters.
3. Failure handling.

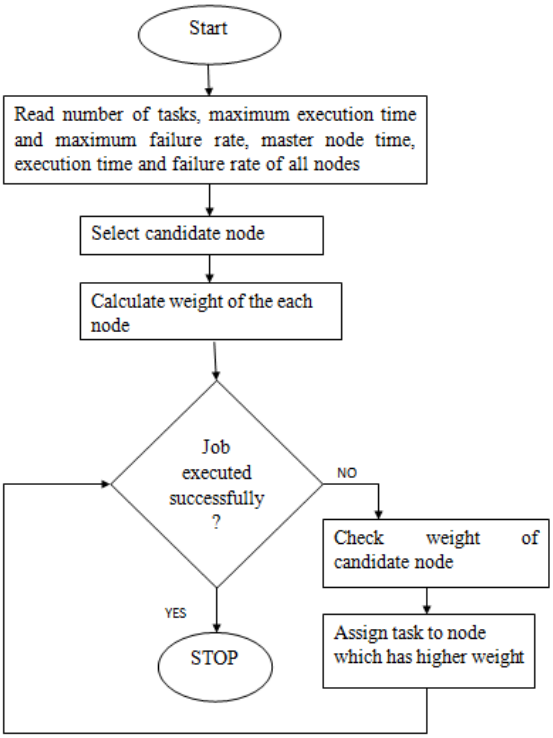

Fig. 4: Flowchart of Proposed Work

5. RESULTS

The proposed work has been implemented on MATLAB platform.

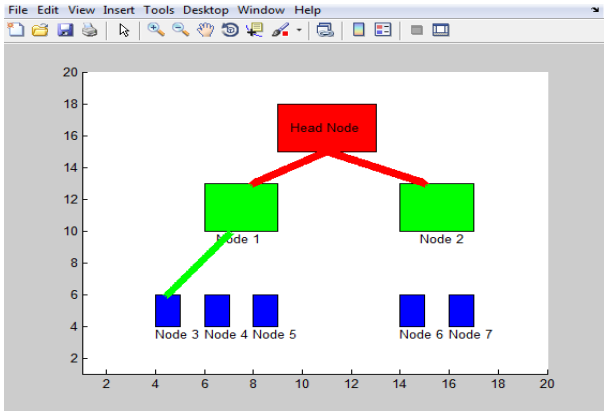

Fig5.1: Task allocation

As shown in figure 5.1, the head node will assign task to its sub nodes. The sub-nodes are responsible to assign tasks to candidate nodes. The sub-node will ask for the number of tasks to be assigned to selected candidate nodes. The candidate nodes when execute their assigned task will revert back to head node. Then the user will enter the time taken by head node to merge result to the tasks the weight of every node is calculated on the basis of applied formula. The candidate nodes have been selected from all available nodes on the basis of formula.

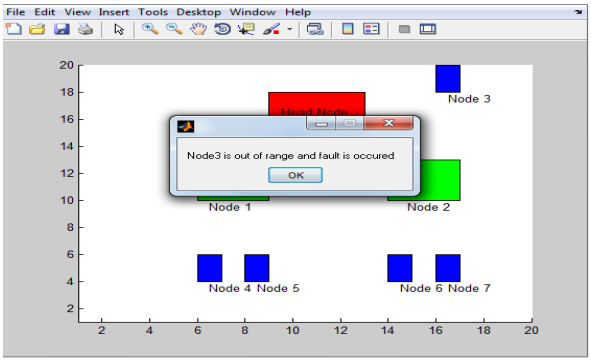

Fig.5.2: Occurrence of fault

Figure 5.2 shows the scenario of fault occurrence. It can be seen that due to mobility of node fault occurred.

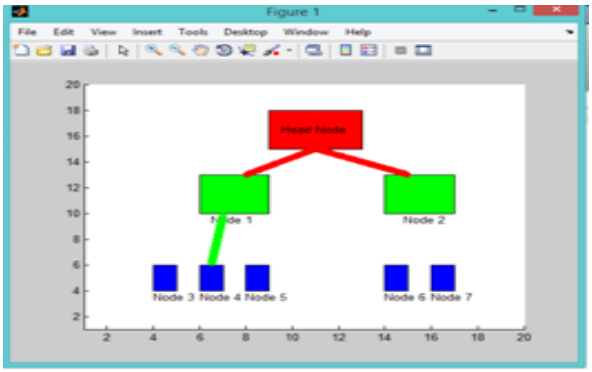

Fig.5.3: Task Re-assignment

Now the sub node will look for the appropriate candidate node by checking weight of each node and selects the one with maximum weight. Figure 5.3 show that task is re-assigned to node with maximum weight.

6. CONCLUSION

As the distributed system is characterized as a dense network of assortment of mobile elements joined by a remote connection, without any fixed support. In this system no central authority is available because of which the system disengagement is exceptionally frequent between the mobile nodes. Because of above reasons possibilities of error in the mobile disseminated network is extremely high. The load is split among the various unstationary nodes to improve the system proficiency and to decrease the job execution time. When the load is not equally split among the mobile nodes, possibility of events of errors will be expanded. The methodology of fault-tolerance is prescribed to decrease the number of error rates in portable dispersed system. The task is allocated among the mobile node with the help of task allocation /assignment 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 weight based technique. The proposed algorithm is based on the number of tasks, failure rate, execution time and master node time. This technique leads to reduction in processing time and reduction in energy consumption.

In future, enhancement in the proposed algorithm can be made to handle certain security attacks in distributed systems; these attacks are denial-of-service attacks which reduces the networks reliability and efficiency. In a denial-of-service (DoS) attack, an attacker endeavors to stop legitimate users from accessing data or services. By targeting your computer and its web connection, or the computers and web of the locations you are trying to use, an attacker could be able to stop you from accessing email, websites, online reports or supplementary services that rely on the altered computer.

7. REFERENCES

[1] Zhongkui Li and Zhisheng Duan, "Distributed tracking control of multi agent systems with heterogeneous uncertainties", International Conference on control and Automation, Hangzhou, pp. 1956-1961, 2013.

[2] Sreedevi R.N, Geeta U.N, U.P.Kulkarni , A.R.Yardi, "Enhancing Mobile Agent Applications with Security and Fault Tolerant Capabilities", International advance Computing Conference, pp. 992-996, 2009.

[3] Asma Insaf Djebbar, Ghalem Belalem , "Modeling by groups for faults tolerance based on multi agent systems", International conference on machine and web intelligence, pp.35-40, 2010.

[4] A. Y. Hamed, "Task Allocation for Maximizing Reliability of Distributed Computing Systems Using Genetic Algorithms", International Journal of computer networks and wireless communications (IJCNWC), 2012.

[5] P K Yadav, M P Singh and Kuldeep Sharma, "An Optimal Task Allocation Model for System Cost Analysis in Heterogeneous Distributed Computing Systems: A Heuristic Approach", International Journal of computer applications, pp. 30-37, 2011.

[6] Shilpa Gambhir ,Er. Sonia Goyal, "Reliable Task Allocation in Distributed Mobile Computing System with random node movement: Replication and Load Sharing Approach", International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE) 2014.

[7] Qin&Ma Kng, Hong He, Hui& Min Song, Rong Deng, "Task allocation for maximizing Reliability of distributed Computing System using Honeybee mating Optimization" ,The Journal of Systems and software, 2010.

[8] A.A. Elsadek & B. E. Wells, "A heuristic model for task allocation in heterogeneous distributed computing systems," International journal of computers and their applications, pp. 1-35, 1999.

[9] V.M. Lo, "Heuristic Algorithms for Task assignment in distributed systems," IEEE Transactions on computers, pp. 1384-1397, 2010.

[10] Vinod Kumar Yadav, Mahendra Pratap Yadav and Dharmendra Kumar Yadav, "Reliable Task allocation in heterogeneous distributed system with random node failure: Load sharing approach", International conference on computing sciences, pp. 187-192, 2012.

[11] S. Ghosh, R. Melhem, D. Mosse, "Fault-tolerance through scheduling of a periodic tasks in hard real-time multiprocessor systems", IEEE Transactions on Parallel and Distributed Systems, pp.272-284, 1997.

[12] G. Rodrıguez, M. J. Martın, P. Gonzalez, and J. Tourino, "Controller/Precompiler for Portable Checkpointing", IEICE Transactions on Information and Systems, pp.408–417, 2006.

[13] Yung- Terng Wang, Robert J.T. Morris, "Load sharing in distributed systems", IEEE Transactions on computers, pp 204-217,1985.

[14] Fouzi Semchedine, Louiza Bouallouche-Medjkoune, Djamil Aissani, "Task Assignment policies in distributed server systems: A survey", Journal of network and Computer Applications, pp. 1123-1129, 2011.

[15] Sanjay Bansal, Sanjeev Sharma, Ishita Trivedi, "A detailed review of fault- tolerance techniques in Distributed System", IJIDCS, pp. 33-37.

[16] George Coulouris, Jean Dollimore, Tim Kindberg Gordon Blair, "Distributed systems- concepts and design", Fifth edition, 2012.