An Approach for Faults Recover occur due to Mobility in Distributed Computing System

N Ramesh Babu

Abstract

Distributed Computing System is heterogeneous in nature. A computer program that runs on distributed system is known distributed program. There are a number of basic problems that likely to amplify the conceptual distance and complicate the design. There are many design issues in distributed computing systems. In this paper, we have discussed various issues of distributed computing system. The main focus is on the mobility of the node. At the end we have proposed methodology to overcome mobility problem based upon pong messages.

KEYWORDS: DCS, task scheduling, mobility, fault tolerance

---

1. INTRODUCTION

A distributed system connected by local networks and physically connected with each others. Distributed computing utilizes a network of many computers, each accomplishing a portion of an overall task, to achieve a computational result much more quickly than with a single computer. Distributed Computing System is heterogeneous in nature. 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]. Grid computing and Cluster computing are types of distributed computing systems. A distributed system consists of a group of independent computers associated through a network and sharing middleware which enables computers to organize their behavior and to share the property of the system so that users identify the system as a single, incorporated computing facility [5]

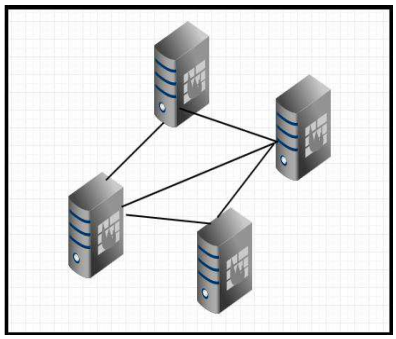

Fig.1.1 Distributed system

1.1. ***Issues in Distributed system:*** The theoretical distance between an analysis and a resultant design is supposed to be kept to a lowest. This improves understandability, traceability and maintainability [24]. For distributed systems, however, there are a number of basic problems that are likely to amplify the conceptual distance and complicate the design process. The main issues in distributed computing are as follow:

1. ***Object distribution***

The distribution of software substance doesn't hold to stand for the sharing of corresponding entity in the real-world. Just because a parcel is in Boston, doesn't mean that the software objects that represent it have to be in Boston. Decisions on how to allocate objects across system should consider other factors like presentation, dependability, security, and fault tolerance [11].

2. ***Complex mappings***

The mapping among concepts in the analysis and individuals in the design is not one-to-one. A single real-world entity can be represented by several software substances. Suppose to enlarge access time, a parcel-tracking system may comprise various software matter for a single parcel and distribute them across the network [5].

3. ***Fragmented behavior***

A software entity may symbolize only a fraction of an entity described in the analysis. It will submit to such partial representations, *as entity fragments.* The object fragments that correspond to an exact real-world thing do not have to be identical as long as they cooperatively summarize the necessary information and behavior. In a client/server environment, customer thing fragments are often very frivolous and correspond with their server counterparts to provide users with the complete functionality [7].

4. ***Transparency***

To help make a dispersed system more unlock, extensible, and fault tolerant, designers attempt to shield users from issues dealing with the actual location of an object and concurrent access, replication, migration, scaling, and failures. This principle is called transparency. Unluckily, technique for achieve clearness further increase the theoretical detachment between investigation and design [8].

5. ***Emergent communications***

Distributed software systems involve many of different kinds of communications. Most of them, however, do not relate directly to communications in the problem domain. Some exist for housekeeping reasons, like replica management and transaction, process synchronization, name resolution, and service binding. Others exist as a result of using exacting communication architecture. In either case, these

emergent communications play a significant role in design, and therefore, cannot be left out of a concept model [6].

6. *Emergent resource sharing*

Distribution can also set up a significant amount of resource sharing that didn't exist in the analysis. In the real-world parcel system, only one person is able to handling a given parcel at a time. In a distributed software system, a lot of different users may be accessing and updating the software items for that parcel at the same time [9].

2. REVIEW OF LITERATURE

In this paper [1], they proposes a novel parallel check pointing algorithm antecedence graph approach for achieving fault tolerance in mobile agent systems. By recording the dependency relation among mobile agents in antecedence graphs and check pointing them to stable storage during the normal computation message transmission, the proposed algorithm can reduce the time latency for a global check pointing procedure significantly. Furthermore, it only forces the minimum number of MAs to take their checkpoints and minimizes the number of blocked mobile agents during identifying, which improves the system performance compared with previous graph based approaches. The quantitative analysis and numerical results reveal that the proposed algorithm has better performance than existing ones and the overheads for proposed scheme are significantly low. These advantages enhance the applicability of fault tolerance approach using antecedence graphs for mobile agent systems.

The future scope of the work includes comparison of the proposed schemes with other existing fault tolerant schemes. In paper [2] Distributed operating systems are still in an early phase of development, with many unanswered questions and relatively little agreement among workers in the field about how things should be done. Many experimental systems use the client-server model with some form of remote procedure call as the communication base, but there are also systems built on the connection model. Relatively little has been done on distributed naming, protection, and resource management, other than building straight- forward name servers and process servers. Fault tolerance is an up-and-coming area, with work progressing in redundancy techniques and atomic actions. In this paper [3] 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 the 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 are to be processed by the processors.

In each phase only one task shall be executing 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 numbers of tasks. Optimal allocation has been obtained along with phase wise optimal costs. In this paper [4] they addressed the distributed tracking control problem for multi-agent systems with heterogeneous uncertainties and a leader whose control input might be nonzero and not available to any follower.

Based on the relative states 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 [5] they represented 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 agent systems must also provide an 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.

3. MOBILITY IN DISTRIBUTED COMPUTING SYSTEM

Fault Tolerance is an important method distributed computing because systems are distributed geographically in this system under different geographically domains throughout the web wide. The most difficult task in grid computing is design of fault tolerant is to verify that all its reliability requirements are meet [7]. The real time distributed systems like grid, robotics, nuclear air traffic control systems etc. are highly responsible on deadline. Any mistake in real time distributed system can cause a system into collapse if not properly detected and recovered 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. A distributed computing system must be fault tolerant. It should be able to continue in its functioning in the presence of faults. Most of the faults are related to dependability.

Further dependability has many sub categories. The distributed computing system is the system in which master nodes divide, the task into multiple tasks, the slaves nodes execute task on the behalf of master node. Task scheduling and execution over large scale, distributed systems plays an important role on 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 increases with

the increasing complexity of their services and policies, system administrators attempt to ensure high quality of services each user requires by maximizing utilization of system resources.

To achieve this goal, correct, real-time and efficient management and monitoring mechanisms are essential for the systems [10]. But, as the infrastructures of the systems rapidly scale up, a huge amount of monitoring information is produced by a larger number of managed nodes and resources and so the complexity of network monitoring function becomes extremely high. There are many cases in which node moves from its position. Due to this mobility fault tolerant problem may occur because communication halts in between the process. To overcome this problem there are many task allocation algorithm available [11].

4. PROPOSED METHODOLOGY

There are heterogeneous and various network environments within the systems needed to be monitored and the nature of managed resources becomes almost dynamic, not static, which forces traditional static centralized and distributed monitoring mechanisms to be unsuitable for the systems. 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 the master and slave architecture through which the task are assigned by the master node to slave nodes and slave nodes execute task as assigned by the master nodes. In such type of architecture, the divide and rule technique is followed, and after executing the task slave nodes will revert back to master nodes.

The major problem in this architecture is of fault, if one slave node get failed the task allocated by master node will not get completed and fault occurred. The task scheduling is the major problem exists in the distributed computing systems. The major task is distributed between various slaves nodes by the master node. The task scheduling algorithms efficiently distributed the task between various slaves nodes, If the tasks are not efficiently divided then the fault will be occurred in the system which will reduce efficiency of the system. In this work, we will enhance the master slave architecture for fault tolerance. The enhancement will be based on the back node and on ping messages, with these two techniques the master node will detect the fault in less amount of time.

This technique will lead to reduce in response time and less message exchange. The proposed idea will be implemented in MATLAB which is widely used in all areas of applied mathematics, in education and research at universities, and in the industry. MATLAB stands for MATrix Laboratory and the software is built up around vectors and matrices.

5. CONCLUSION

A distributed system consists of a group of independent computers associated through a network and sharing middleware which enables computers to organize their behavior

and to share the property of the system so that users identify the system as a single, incorporated computing facility. There are many issues in distributed computing systems which are responsible for the degradation of the system. In this paper main focus is on mobility of the node. Due to mobility efficiency of the system decreases. To overcome this problem a novel technique has been discussed based upon ping messages.

REFERENCES

[1] George Coulouris, Jean Dollimore and Tim Kindberg , Distributed Systems : Concepts and Design, Addison-Wesley,Pearson Education 3rd Edition 2001.

[2] Shan Zhang and Jian-ping Wu , "Construction of Distributed and Heterogeneous Data Sharing Platform" , 2009 International Conference on Web Information Systems and Mining

[3] Vinod Kumar Yadav, Mahendra Pratap Yadav and Dharmendra Kumar Yadav , "Reliable Task Allocation in

[4] 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

[5] Jinho Ahn, "Lightweight Fault-tolerance Mechanism for Distributed Mobile Agent-based Monitoring" IEEE,2008

[6] Tome Dimovski, Pece Mitrevski, "Connection Fault-Tolerant Model for Distributed Transaction Processing in

Mobile Computing Environment" ITI 2011 33rd Int. Conf. on Information Technology Interfaces, June 27-30, 2011, Cavtat, Croatia

[7] Rajwinder Singh, Mayank Dave, "Using Host Criticalities for Fault Tolerance in Mobile Agent Systems, 2nd IEEE International Conference on Parallel, Distributed and Grid Computing, 2012

[8] Asma Insaf Djebbar, Ghalem Belalem , "Modeling by groups for faults tolerance based on multi agent systems",IEEE,2010

[9] 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

[10] ANDREW S. TANENBAUM and ROBBERT VAN RENESSE, "Distributed Operating Systems", 2006

[11]Dr. Kapil Govil, "A Smart Algorithm for Dynamic Task Allocation for Distributed Processing Environment" International Journal of Computer Applications (0975 – 8887) Volume 28– No.2, August 2011