Fault Tolerant Scheduling in Cloud Systems Using Virtualization and Load Balancing

Divyani N. Deshmukh
SGBAU, Amravati
Maharashtra

Dr. Y. M, Kurwade
SGBAU, Amravati
Maharashtra

Dr. V. M. Thakare
SGBAU, Amravati
Maharashtra

ABSTRACT

Scheduling and Fault tolerance reliability of a cloud system is very important aspect. Several models are proposed on fault tolerance since last several decades. This paper focused on five existing fault tolerance and scheduling techniques for cloud systems. Also this paper, presents a method for fault tolerance and scheduling for cloud systems using Virtualization and Load Balancing. A Dynamic resource scheduling performed by load balancing techniques also improves the performance of the system. Implementation complexity of proposed system is low as compared with other strategy.

Index Terms –Fault tolerance, Load balancing, Scheduling and Virtualization

1. INTRODUCTION

For cloud structure, fault tolerance for real-time tasks to be made a common challenge now a day. The large cloud data centres borrow a high resources failure chance due to the accretion functionality and complexity of the large systems. With the increasing complexity, the eventuality of hardware faults and software failures increases. However, a distributed control system is a kind of hard real-time system, in which the consequences of not executing a task before its deadline may be catastrophic [1]. In some hard real-time systems, such as computer integrated manufacturing and industrial process control, a number of tasks are radically and executed in order to invoked collectively accomplish a common mission/ function and each of them must be completed by a certain deadline.
Failure to entire such a task in time may lead to a demure accident [2]. Fault-tolerance by primary-backup replication, which indicates that a primary task includes zero, one, or multiple backup tasks, is an important credibility enhancement mechanism. In general, primary and backups are conjointly referred to as replicas [4].The heterogeneous or the g

Geographically widely distributed system, as thegreater differences of hardware models, the software versions or the communication delay, the start time of different replicas is difficult to control,which adds complexity and even makes the fault-tolerant impossible in the actual system environment. Whereas in passive replication mode, if the primary replica fails, the task will be assigned to the resource that can achieve high reliability and low communication cost, there by resulting in better availability for the task [5].

This paper introduces different fault tolerant and scheduling strategies such as BNPRMFT, NEFT, Energy-efficient fault-tolerant scheduling with reliability goal algorithm, Survivability-aware Fault-tolerant Scheduling Algorithm and Checking Available Time algorithm and Eliminate idle time algorithm. These algorithm gives better outcome but some limitations such as the algorithm can’t work when multiple host failure on cloud. Also some have poor response time. These limitations are overcome by using virtualization and load balancing technique for scheduling cloud tasks and proper utilization of cloud resources. This archives a proper fault tolerance scheduling.

2.BACKGROUND

Fault tolerance scheduling are designed for a kind of faults but there is no algorithm which can tolerate both hardware faults and software faults [1],[3].Backward non-pre-emptive RM (BNPRMFT), which can tolerate both hardware faults and software Faults. BNPRMFT execute primary copies and lower the runtime overhead and can obtain a higher success rate in [1].Checking Available Time (CAT) algorithm and Eliminate idle time (EIT) is being evolved for guarantying that either the primary or alternate version of each critical task to be completed in time [2].Various fault-tolerant workflow scheduling algorithm are evolved for tolerating hardware and software fault. When corresponding primary copy fails due to faults, the backup copy of a task is executed only. In executing primary copies and lower the runtime overhead of the algorithm NEFT can obtain a higher success rate [3].In embedded system dynamic voltage and frequency scaling is well known energy consumption technique. Energy-efficient fault-tolerant scheduling with a reliability goal (EFSRG) algorithm to reduce the energy consumption while satisfying the reliability goal based on an active replication scheme [4]. The accumulated reliability is introduced for evaluating the good resources reliability in a period of time. Survivability-aware Fault-tolerant Scheduling Algorithms (SFTS) is designed to schedule tasks from the originating node resource to the executing node resource. Also this algorithm can tolerate multiple failures for critical tasks with survivability demand in heterogeneous system[5].

The rest of the paper is organized as follows. In this paper, ***Section II*** gives us background details, ***Section III*** provides work which is done previously, ***Section IV*** gives idea about existing technology, in ***Section V*** analysis and discussion about techniques is carried out, proposed methodology is explained in ***Section VI***, Possible outcomes and Result is described in ***Section VII***, ***Section VIII*** concludes the paper. Finally, ***Section IX*** described future scope of the paper.

3.PREVIOUS WORK DONE

Scheduling and Fault tolerance reliability of a distributed system is an important aspect. Several models are evolved for fault tolerance in last several decades.

Jian Wu et.al.(2003)[1] proposed Checking Available Time (CAT) algorithm and Eliminate idle time (EIT) is being evolved for guarantying that either the primary or alternate version of each critical task to be completed in time. This algorithm is enhanced further to prevent early failures from triggering failures in the subsequent job executions, thus improving efficiency of processor usage.

Xiangruet.al.(2016)[2] proposed Survivability-aware Fault-tolerant Scheduling Algorithms (SFTS) is designed to schedule tasks from the originating node resource to the executing node resource. This architecture adopts the sequential approach for tolerating multiple failures for critical task with survivability.

Liu et.al.(2016)[3] proposed NEFT algorithm which can tolerate both hardware faults and software Faults. This architecture adopts the star topology communication model.

Liang et.al.(2017)[4] worked on backward non-pre-emptive RM (BNPRMFT), which can tolerate both hardware faults and software Faults. This architecture adopts the star topology communication model where the lots lot ofprocessors interconnected by a certain bandwidth limited interconnection network.

Cheng Xuet.al.(2017)[5] proposed energy-efficient fault-tolerant scheduling with a reliability goal (EFSRG) algorithm to reduce the energy consumption while satisfying the reliability goal based on an active replication scheme. This architecture adopts the Dynamic voltage and frequency scaling technique in embedded systems

4. EXISTING METHODOLOGIES

4.1Checking Available Time algorithm and Eliminate idle time Algorithm:-

Fault-tolerant algorithm and schedulability analysis are being defined by two more ideas to improve the percentage of the successful primaries. This method uses a fixed priority-driven pre-emptive scheduling scheme to reallocate time intervals to the alternates. This architecture adopts the sequential approach where one by one algorithm is being implemented to achieve reliability. Earliest Deadline First algorithm which is used to serve time interval for the alternates Backwards RM algorithm proposed which assigns priorities to tasks according to the rate monotonic rule. Checking Available Time i.e. CAT algorithm which is use to eliminate the wasteful execution of primaries. After that proposed Eliminating Idle Time i.e. EIT Algorithm which is used to chooses an alternate to execute when the processor is about to be idle. Based on this fault-tolerant and scheduling mechanism both CAT and EIT algorithm comes together which decreases the wasted processor time and enhances the percentage of successful primaries. There are "n" time intervals that have been allocated for alternates by the offline backward-RM algorithm, which is determined by following expression:

$$AT_{ij} = (v_{ij} - t) - \sum_{i=1}^{l}$$

4.2Survivability-aware Fault-tolerant Scheduling Algorithm:-

Survivability-aware Fault-tolerant Scheduling Algorithms (SFTS) is designed to schedule tasks from the originating node resource to the executing node resource. This architecture adopts the sequential approach for tolerating multiple failures for critical task with survivability. Survivability is the ability that systems timely complete the critical tasks under attacks, failures or accidents. When a task arrives, scheduling of the particular task is held by system model and task model of scheduling. In this scheduling model to achieve survivability-aware scheduling, passive task replication mode is being adopt. For the resource Pm the execution cost that the task ts is scheduled to it can be formulated as follows:

$$ec(t_s, p_n) = \frac{k(p_{s,n})+1}{\mu_{s,n}} + \sum_{i=1}^{s-1} \frac{k(p_{i,n})}{\mu_{i,}}$$

4.3 NEFT Algorithm:-

This architecture adopts the star topology communication model where the lots lot of processors interconnected by a certain bandwidth limited interconnection network. In a Distributed control system (DCS), there are lots of control loops running on processors which communicate each other. To decrease the number of pre-emption and lower the runtime overhead of the algorithm, non-pre-emptive Earliest Deadline First (EDF) is used to schedule primary copies which are assigned to the same processor, and the backward non-pre-emptive EDF is used to schedule backup copies and to calculate notification times of tasks. NEFT can obtain a higher success rate in executing primary copies and lower the runtime overhead of the algorithm. How primary copy is assigned to a processor, load of backup copies assigned to (Prm) whose corresponding primary copies are assigned to (Prj) the load of backup copies assigned to Prl, and ub (Pri)is define in this:

$$u_p(\Pr_j) = \sum_{\tau_i.t^p.Pr=\Pr_j} \frac{\tau_i.t^p.C}{\tau_i.T}.$$

,

$$u_b(\Pr_m|\Pr_j) = \sum_{\substack{\tau_i.t^b.Pr=\Pr_m \\ \tau_i.t^p.Pr=\Pr_j \\ m\neq j}} \frac{\tau_i.t^b.C}{\tau_i.T}.$$

4.4 BNPRMFT Algorithm:-

Both hardware and software faults are tolerated by the fault tolerance scheduling algorithm based on backward non-pre-emptive RM. BNPRMFT can obtain a higher success rate in executing primary copies and lower the runtime overhead of BPRMFT. This architecture adopts the star topology communication model where the lots lot of processors interconnected by a certain bandwidth limited interconnection network. In Distributed control system a scheduling algorithm must determine when and where a task is executed.In order to realize fault-tolerance, two copies of each task are employed, which are assigned to two different processors.Only when a primary copy fails, the corresponding backup copy assigned to another processor is executed. The passive backup copy technique is employed here. How primary copy is assigned to a processor, the load of backup copies assigned to (Prm) whose corresponding primary copies are assigned to (Prj) is formulated as follows

$$u_p(\Pr_j) = \sum_{\tau_i.t^p.Pr=\Pr_j} \frac{\tau_i.t^p.C}{\tau_i.T}.$$

,

$$u_b(\Pr_m|\Pr_j) = \sum_{\substack{\tau_i.t^b.Pr=\Pr_m \\ \tau_i.t^p.Pr=\Pr_j \\ m\neq j}} \frac{\tau_i.t^b.C}{\tau_i.T}.$$

4.5 Energy-efficient fault-tolerant scheduling with a reliability goal algorithm:-

This architecture adopts the Dynamic voltage and frequency scaling technique in embedded systems and dynamically scaling down the voltage of a chip which has been developed to achieve energy-efficient optimization. Energy-efficient scheduling with a reliability goal algorithm, to reduce energy consumption while satisfying the reliability goal of a DAG-based parallel application on heterogeneous embedded systems without using fault-tolerance.An energy-efficient fault-tolerant scheduling with a reliability goal (EFSRG) algorithm, to reduce the energy consumption. The problem is solved by dividing it into three sub-problems i.e. prioritizing tasks, satisfying reliability goal, and reducing energy consumption. Failure rate of the processor with the maximum frequency, then the failure rate of the processor with the frequency f is calculated as follows:

$$\lambda_{k,v} = \lambda_{k,\max} \times 10^{\frac{d(f_{k,\max}-f_{k,v})}{f_{k,\max}-f_{k,\min}}}$$

5. ANALYSIS AND DISCUSSION

This section describes brief analysis of the above five method. The table shows the comparison between five existing methods and also shows advantages and disadvantages of five methods.

Fault Tolerant and Scheduling Techniques.	Advantages	Disadvantages
Checking Available Time algorithm and Eliminate idle time Algorithm	➢ Reduces execution time. ➢ Decreases the wasted processor time and enhances the percentage of successful primaries. ➢ chooses an alternate to execute when the processor is about to be idle	➢ This method cannot handle alternate execution which may fails due to processor failure. ➢ This method does not provide high efficiency ➢ The proposed algorithm can't work when to add specific parameters in the each task set.
Survivability-aware Fault-tolerant Scheduling Algorithm	➢ It is easy-to-deploy and low overhead in non-pre-empting scheduling ➢ It promotes the system better survivability ➢ It strives to choose perfect node resources and scheduling paths.	➢ It just concentrates on the critical task rather than the entire task. ➢ It has shorter response time of critical task. .
NEFT Algorithm	➢ It obtains a higher success rate in executing primary copies and lowers the runtime overhead. ➢ It can tolerate both hardware faults and software Faults. ➢ It decreases the number of pre-emption greatly.	➢ Does not schedule back up copy as efficiently as primary copy. ➢ Can't work when multiple back up copy schedule on a distributed system.
BNPRMFT Algorithm	➢ It can obtain a higher success rate in executing primary copies and lower the runtime overhead. ➢ It gives high control performance to execute many primary copies. ➢ It can	➢ It does not schedule back up copy as efficiently as primary copy ➢ It can't work when multiple back up copy schedule on a distributed system.

	decrease the number of pre-emption greatly.	
Energy-efficient fault-tolerant scheduling with a reliability goal algorithm	➢ It is used to reduce the energy consumption. ➢ It implements efficient fault tolerance without increasing time complexity. ➢ It generates less energy consumption.	➢ It does not provide energy-efficient fault-tolerance scheduling. ➢ It does not consider considering reliability goal and timing constraint.

Table 1:- Comparisons between different Fault Tolerant and Scheduling Techniques.

6.PROPOSED METHODOLOGY

This section provides the proposed methodology which has used a concept of Virtualization technique to make the system fault tolerant and for dynamic resource scheduling load balancing techniques are proposed.
First establish a remote server group, hypervisor such as VMW are can double a centres virtualized IT structure. Using v Replicator service between the operating end and the remote end, a real-time application copy can be created from the virtual machine at the operating end to ESX host storage in the remote end. This can allow for disaster recovery in different places. V replicator operates with virtual machines, monitors virtual machine disk file data changes, and after a full disk data copy operation is completed, every other five minutes automatically copies different data between the disk data and the disaster recovery end. When the operating server has a service disruption, v replicator automatically performs a failover operation for virtual machine backup, whose disaster recovery end is on standby status. Backup machine data and settings exactly match the virtual machine source, so after starting, the application can be immediately taken over and restarted, providing service for final users.
Proposed dynamic resource scheduling uses load balancing techniques. Here the master node dynamically allocates tasks through slave nodes to achieve balanced loads. After load balance information is collected from slaves, dynamic configuration can be triggered. Dynamic configuration is just an operation in which data is migrated from one slave node to another. Migration just means the data is copied from the slave with the higher load to another slave with a lower load, and that the data is deleted from the original, higher load slave.
Dynamic resource scheduling performs three crucial resource-related operations:

- ➢ It calculates the demand of resources that each VM should request based on the reservation and shares settings and constraints for VMs, as well as resource pool nodes.
- ➢ It does initial placement of VMs on to hosts automatically.
- ➢ It suggests and performs live VM migrations to do load balancing across hosts in a dynamic environment when the VM's resource demands vary during a period of time.

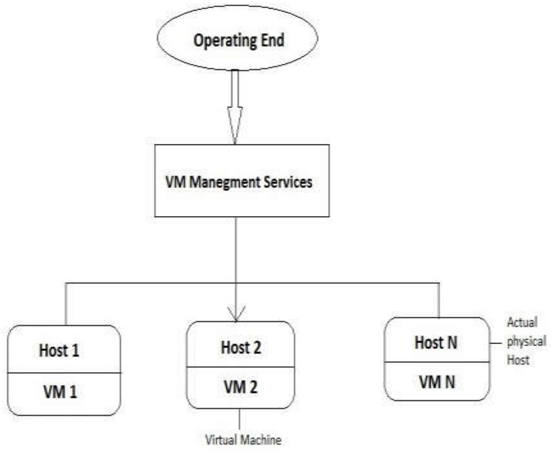

Figure 1:- VM System Structure.

7.OUTCOME AND POSSIBLE RESULT

As Dynamic resource scheduling performed by load balancing techniques this improves the performance of the system. Optimization goal makes the utilization of a resource reach to the average value. System complexity is low as compared with other strategy.

8. CONCLUSION

In this paper, the problem of Virtualization technique is studied to make the system fault tolerant and for dynamic resource scheduling i.e. CAT and EAT algorithm, SFTS algorithm, NEFT algorithm, Backward non-pre-emptive RM algorithm and energy-efficient fault-tolerant scheduling with a reliability goal algorithm. The proposed methodis lower overhead in non-pre-empting scheduling. It can decrease the number of pre-emption greatly. By integrating easy but elegant heuristics it makes significant performance improvement. It has longer response time, because its objective is to achieve load- balance of the system, which achieve resource allocation.

9.FUTURE SCOPE:

The proposed method which has used a concept of Virtualization technique does not consider reliability goal and timing constrain together for fault tolerance scheduling. It does not address the issue of efficiency and scheduling backup copy on a Distributed control system also issue of shorter response time. Future study tries to overcome this issue.

REFERENCES

[1] Liang Lie, Liu Huai,"A Fault-Tolerant Scheduling Algorithm for Distributed Control System Based on Backward Non-Preemptive RM.",Vol. x, Issue no. x, PP. 2540-2545 ,January-2017.

[2] Ching-Chih Han, Member, IEEE, Kang G. Shin, Fellow, IEEE, and Jian Wu, Student Member, IEEE,"A Fault-Tolerant Scheduling Algorithm for Real-Time Periodic Tasks with Possible Software Faults." *IEEE Transaction on computers,*Vol. 52, Issue no. 03, PP 362-372, March-2003.

[3] Liu Huai, Huang Jianxin,"A Fault-Tolerant Scheduling Algorithm for Distributed Control System with Possible Hardware and Software Faults." *IEEE Transaction on Distributed Systems,* Vol. x, Issue no. x, PP2646-2651, April-2016.

[4] GuoqiXie, Member, IEEE, Yuekun Chen, Xiongren Xiao, Cheng Xu, Renfa Li, Senior Member, IEEE, andKeqin Li, Fellow, IEEE. "Energy-efficient Fault-tolerant Scheduling of Reliable Parallel Applications on Heterogeneous Distributed Embedded Systems"*IEEETrasaction on Sustainable computer,* Vol. x, Issue no. x, PP 1-16,January 2017.

[5] Wenyan Cui, XiangruMeng, Yakun Zhang, and ZhiyuanZhao"Survivability-aware Fault-tolerant Scheduling Using Primary-backup Approach in Heterogeneous System"*IEEE-2016*, Vol. x, Issue no. x, PP 2823-2828,April-2016.