Fault Tolerant and Scheduling in Cloud Systems Using Earliest Ending-Time First

[1] Divyani N. Deshmukh, [2] Dr. Y. M. Kurwade, [3] Dr. V. M. Thakare

[1]ME student, [2]Professor, [3]Professor,
[1]Department of Computer Engineering, [2]Department of Computer Engineering, [3]Department of Computer Engineering,
[1] SGBAU, Amravati, India, [2] SGBAU, Amravati, India, [3] SGBAU, Amravati, India

Abstract: Cloud computing has mainly worried about scientific computing area. Fault tolerance plays an eventful role for amending the probity of a system. This paper focused on five perspectives for cloud system to schedule task, cloud resources allocation and finding energy consumption by physical machines. Earliest ending time first algorithm is mainly used for scheduling the task as per the priority. It uses Queue data structure for scheduling and storing recent task. It also keeps control on physical devices to find energy consumption.

Index Terms **- Digital Terrain analysis, Petri nets, Cloud service provider, Real-time system, Primary-backup copy..**

I. INTRODUCTION

Recently, for resolving whole computation-intensive and data-intensive applications, cluster computing has to be made a glamorous computing schema. The probity of the system to be made foremost key while the stability of cluster with tens of thousands of processors is threatened constantly by a larger number of hardware and software failures, such as network, memory, processors and operating systems [1]. Fault-tolerant real-time scheduling provides a method of combining timing requirements and faults tolerance for real-time systems [2]. To amend the credibility of multiple earth-observing satellites, especially in emergent scenarios such as obtaining photographs on battlefields or earthquake sector, it plays an eventful role. Fault tolerance to be able to carry out through scheduling approaches. Scheduling has been

Presumed as an efficient approach, to accomplish high performance in satellite imaging, [3]. In scientific computing area cloud computing has been increasingly concerned. To hamper the widespread adoption of cloud computing credibility has become one of the largest block [4]. To soft errors in modern high tentative systems, aggressive technology and voltage scaling has drastically accretion their susceptibility. At the grand scale of cloud computing, it is clear that soft error surrounded errors will take place afield more often, but it is unclear as to how to emphatically apply current error detection and fault tolerance techniques in scale [5]. Therefore, the fault-tolerance has beseemed a compulsory integrant of the credible computations.

This paper introduces distinct fault tolerant and scheduling strategies such as Fault-Tolerant Granularity Model based Scheduling Algorithm, AJD Algorithm, Static and Dynamic scheduling Algorithm, Check point fault tolerance RM Algorithm, Fault Tolerant Satellite Scheduling Algorithm. These algorithms gives better outcome but some limitations such as the some algorithm cannot provide fault tolerance reliability. It increases the overheads of kernel queue operations and does not enhance scheduling accuracy. Also on multiple host failure these algorithm cannot work properly. These limitations are overcome by using LTG combining with EETF for scheduling cloud tasks and proper resources utilization of cloud. LTG can undertake that all deadlines are met provided that the total CPU utilization is not more than 100%. This archives a proper fault tolerance scheduling for cloud systems.

II. BACKGROUND

Fault tolerance software is to enhance the stability and robustness of scientific application. Diskless check pointing techniques have been studied to improve system-level fault tolerance mechanisms. Fault-tolerant is being handling by scheduling the failing task which dividing all the failing data into several partitions according the calculating scale of failure [1].ADJ is used to schedule the task also compute average queuing time in each user. Petri techniques are used to analyze and validate the correctness of proposed method [2].

Fault-tolerant is demanded in data storage, transmission and computing. Cloud service provider (CSP) perspective is considered for soft error resiliency. For reliability, performance, and energy unified resource allocation and fault tolerant scheduling framework are introduce[3].Check-Point based Fault-Tolerant RM (CP-FTRM) scheduling policy gives feasibility and efficiency in the real- time system. It is used to decrease task error recovery overhead [4].Novel fault-tolerant satellite scheduling algorithm i.e. FTSS to improve the resource utilization, it will apply the overlapping technology. To improve the resource utilization, it applies the overlapping technology that includes primary backup copy overlapping (PB overlapping) and backup-backup copy overlapping (BB overlapping) [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.

III. PREVIOUS WORK DONE

Fault tolerance scheduling for parallel terrain analysis has becomes a common challenge for cloud computing. Many algorithms is proposed to outbrave the difficulty of failing task, deadline handling, soft error resilient, time redundancy, overlapping and merging the task.

Engelmann et al.[2014][1] proposed diskless check pointing of FFTs to super-scale to solve fault tolerance difficulty on huge distributed systems. The solution is based on the checkpoint data of dynamic fault-tolerant granularity model.

Ghosh et al. [2015][2] proposed a stochastic sub model by extending the stochastic Petri nets. These works provide helpful suggestions about making fault tolerant strategy. This architecture adopts the First in First out strategy.

Yue Gao et.al. [2014][3] Proposed cloud service provider i.e., CSP to derive high error coverage and fault tolerance. This architecture adopts the First in first out methodology for message communication. In this architecture static and dynamic scheduling are responsible for handling workload, failure detection, fault tolerance and fault detection.

Min et.al.[2016][4] proposed effective technology to realize system dependability, especially in real-time system. In this seL4 microkernel provides a capability-based access-control model to isolate software components, and also to enable authorized system calls and Inter-Process Communication. This architecture adopts the first in first out approach as it uses the event queue. In this significant amount of research on implementing on real-time scheduler, time management related to real-time systems and fault-tolerant real-time scheduling policies.

Tsuchiya et al. [2017][5] proposed a technique in which two copies of each task are concurrently executed with different start times. A fault tolerant scheduling algorithm makes the two copies of a task which executed simultaneously to improve schedulability.

IV. EXISTING METHODOLOGY

A. Fault-Tolerant Granularity Model based Scheduling Algorithm:

This Fault-tolerant algorithm stores all the Executes states and data directories of each thread based on diskless checkpoint. In this all the precedence relations and file directories of processor data block, and recovery states of each processor are periodically stored in computing nodes. In this architecture 2 masters are being there, where master2 is a redundancy of master1. Master1 is used for failure detection and failure recovery of computing nodes. In this checkpoint data is regularly saved on another processor under the passive copy mode. The master node is responsible for the data distribution of slave processors. Each slave processor will receive a file directory of Net File System (NFS) after the data distribution from master node. By this architecture load balancing is achieved and the difference of computing time of each processor will not be too wide. Confirmation message can be used as a condition to judge whether the processor fails or not. The status of, which schedules any failed task and check overhead comparisons of different data determined by the following expression:

$$FG = \frac{Nerr \times D}{n - Nerr} \qquad Per = \frac{Rec}{Com}$$

,

B. **AJD Algorithm**:

Dynamic task scheduling AJD algorithm used to analyze and compute the task scheduling scheme in order to ensure that task can function in time. This algorithm starts from the initial state of model, after that gradually analyzes and solves the next reachable state. This algorithm can terminated, because fault tolerant scheduling model does not have the deadlock, so the state space of the model is limited. This algorithm dynamically compute the execution mode and scheduling scheme for task based on the attributes of state. The following expression states that, computing the unexecuted workload of VM and Execution time.

$$\begin{aligned} &\mathrm{Wd}(\mathrm{vm}_f) \\ &\quad = \max\left\{0,\ \mathrm{dw}_f + \sum_{\mathrm{Jd}_{g,h} \in \mathrm{JV}(\mathrm{vm}_f)} \mathrm{Jd}_{g,h} - \mathrm{ts}_f \times T_{\mathrm{now}}\right\}. \end{aligned}$$

$$ae\left(\mathrm{TK}_{i,j}^{B}\right) = \begin{cases} 0, & \mathrm{TK}_{i,j}^{P} \text{ succeeds} \wedge st\left(\mathrm{TK}_{i,j}^{B}\right) = \text{passive} \\ \frac{\mathrm{Jw}_{i,j}}{\mathrm{ts}_f}, & \mathrm{con}_1 \vee \left(\mathrm{TK}_{i,j}^{P} \text{ fails}\right) \\ pt\left(\mathrm{TK}_{i,j}^{P}, \mathrm{vm}^{P}\right) - \frac{\mathrm{dw}_f}{\mathrm{ts}_f}, & \text{else.} \end{cases}$$

C. Static and Dynamic scheduling Algorithm:

Fault tolerance cloud scheduling having two phases as static scheduling and dynamic scheduling. In static scheduling, cloud service provider performs resource allocation for each user, establish level of effort devoted to the error detection and fault tolerance. Also CSP maps each user to a server and temporal schedule is generated. This scheduling can easily recover a single task crash by gathering output from the replica. Static scheduling only serve as a reference, it can't handle impulsive faults though it can handle by dynamic scheduling. Dynamic scheduling manage output comparisons and initiate re-execution when apropos. It also initiate dynamic allocation and task migration when apropos. This algorithm anticipated deadline violation and creates opportunities for future execution.

$$Total\_Energy = \sum_{x=1}^{M} \left(\sum_{t=1}^{t_{Max}} \left(P_{static}^{x}(t) + P_{dynamic}^{x}(t) \right) \right).$$

D. C-HEFT Algorithm

The C-HEFT algorithm is extended using standard HEFT algorithm to produce efficient cluster based task scheduling and mapping of heterogeneous resources. Workflow-mapper, workflow-engine, job-scheduler and failure-monitor these four major components are in the system architecture. In this a single execution site which consists of multiple VMs are considered. The SWf clustered tasks are executed remotely on separate worker nodes. The workflow-mapper generates an executable workflow from an abstract-workflow provided by the SWf user. The workflow-engine executes the single-clustered job, if its parent jobs have completed their execution. The job-scheduler manages individual clustered jobs and execution on remote resources. Failure-monitor gathers the information such as resource id, failed task id and job id of clustered jobs which failed during execution, and these information are provided to the job-scheduler for resubmission. The job-wrapper in the execution site extracts tasks form clustered jobs and executes it on the worker nodes. Each task t is executed by determining its parent tasks, more accurately the one that completes the communication at the latest time. The task t of the earliest start time (EST) and earliest finish time (EFT) are defined as follows.

$$EST(t_i) = \begin{cases} 0, & \text{if } t_i = t_0 \\ max_{t_p \in P_i} EST(t_p) + e_p, & \text{otherwise.} \end{cases}$$

$$EFT(t_i) = EST(t_i) + e_i$$

E. Dynamic Fault Tolerant Scheduling Algorithm

In this FTSS, to better the resource usage, it incurred the overlapping technology that includes primary-backup copy overlapping and backup-backup copy overlapping. For task merging, it integrates the overlapping mechanism with FTSS. FTSS consider observation resolution, resource utilization, schedulability and fault tolerance. To make a primary copy and its corresponding backup copy be allocated successfully, primary copy should be allocated successfully, as primary copy should be executed as early as possible to save more time slots for backup copy to be allocated before its deadline. Time complexity of primary allocation can evaluate in FTSS efficiently. Time required to start task from its previous task is done by following expression:

$$p^{XP}_{i-1,i,j} = o_j + \frac{|\theta^X_{i-1,j,k} - \theta^P_{i,j,q}|}{s_j} + as_j + b_j.$$

$$r^P_{ij} = \max\{p^{XP}_{i-1,i,j} + ft_{i-1,j}, a_i\}$$

V. Proposed Methodology

This section provides the proposed methodology here Least Time to Go (LTG) is combine with Earliest Ending-Time First (EETF) algorithm for better task scheduling, proper cloud resources allocation and finding energy consumption by physical machines. LTG is a dynamic priority scheduling algorithm which places tasks in a priority queue. Whenever a scheduling event occurs the queue will be searched for the task closest to its deadline. This task is the next to be scheduled for execution. With scheduling periodic tasks that have deadlines equal to their periods, LTG has a utilization bound of 100%. That is, LTG can guarantee that all deadlines are met provided that the total CPU utilization is not more than 100%. Compared to fixed priority scheduling techniques like rate-monotonic scheduling, LTG can guarantee all the deadlines in the system at higher loading.

To find the energy consumption of all physical machines EETF algorithm is used. The main idea behind the ending-time first (ETF) algorithm is that, first; the virtual machines are sorted in ascending order by their end-times and based on the maximum load of overlapping time periods to find the minimum number of physical machines needed. Then virtual machines are allocated base on the physical machine load and, ultimately, the energy consumption of all the physical machines can be found.

Algorithm

Input: VM requests betoked by their (required VM type IDs, start-time, ending-time, requested capacity), the number of the request i is denoted as Ni.

Output: IDs of PMs for all VMs, the number of the needed PMs, the total energy consumption.

Initialization: allocating an ID to each PM.

1. sort the virtual machine in ascending order of their end-time;
2. for i = from1 to n do
3. d = 0;
4. if they are not overlapped or overlapped but still can share resources of an PM do;
5. allocate i to the PM d;
6. else;
7. start a new PM; d = d + 1; allocate i to PM d;
8. end;
9. end for

VI. Conclusion

Cloud applications are generally enormous scale which consists of distributed nodes. This paper focused on the energy consumption of all physical devices and scheduling the cloud resources. In this, many different fault tolerant and scheduling techniques for cloud are proposed i.e. Fault-Tolerant Granularity Model based Scheduling Algorithm, AJD Algorithm, Static and Dynamic scheduling Algorithm, Check point fault tolerance RM Algorithm and Fault Tolerant Satellite Scheduling Algorithm. The proposed method improves the performance of the cloud system by sorting the task in ascending order by their end-times and load of overlapping time periods to find the minimum number of physical machines needed. Proposed enforcement method can assured the reliability and real-time of the system.

VII. Future Scope

The proposed method of a cloud system will describe the failure rate of system, which can compute the stability of each state. The proposed method will extend fault tolerant scheduling model to tolerate multiple satellite failure. Also it decreases the overheads of kernel queue operations by using effective data structure such as bitmap. Future study tries to overcome this issue.

References

[1] Xiaodong Song, Wanfeng Dou, Guoan Tang, Kun Yang,Kejian Qian,"A Fault Tolerance Scheduling Algorithm for Parallel Terrain Analysis", IEEE-2012, Vol. x, Issue no. x, PP 81-85, May-2012.

[2] Xiaomin Zhu, Member, IEEE, Jianjiang Wang, Xiao Qin, Senior Member, IEEE, JiWang, Zhong Liu, Member, IEEE, and Erik Demeulemeester, "Fault-Tolerant Scheduling for Real-Time Tasks on Multiple Earth Observation Satellites.",IEEE Transaction on Parallel and Distributed System, Vol. 26, Issue no. 11, PP 3012-3026,November-2015.

[3] Guisheng Fan, Liqiong Chen, Huiqun Yu, Senior Member, IEEE, and Dongmei Liu, "Modeling and Analyzing Dynamic Fault-Tolerant Strategy for Deadline Constrained Task Scheduling in Cloud Computing." IEEE Transaction on System, Man, Cybernetics: Systems,Vol. x, Issue no. x, PP 1- 15, December-2017

[4] LibinXu, YuebinBai, Kun Cheng, LingyuGe, DanningNie, Lijun Zhang, Wenjia Liu. ,"Towards Fault-Tolerant Real-Time Scheduling in the seL4 Microkernel.",IEEE 18th International Conference on High Performance Computing and Communications., Vol. x, Issue no. x, PP 711-718,June 2016.

[5] YueGao, Sandeep Gupta, Yanzhi Wang, Massoud Pedram,"An Energy-Aware Fault Tolerant Scheduling Framework for Soft Error Resilient Cloud Computing Systems", IEEE Transaction on computer, Vol. x, Issue no. x, PP 1-06, June-2014. **[6]** Government Regional Library, Nashik Road.