Fault Tolerant Scheduling in Cloud Systems Using Genetic Algorithm

Divyani N.Deshmukh **Dr. Y. M. Kurwade** **Dr. V. M. Thakare**

Abstract- **For real-time tasks fault tolerance becomes a common challenge for cloud. A high resources failure probability due to the increased functionality and complexity of the large systems, incurred by the large cloud data centers. This paper focused on five existing fault tolerance and scheduling techniques for cloud systems. This paper attends a method for fault tolerance and scheduling for cloud systems using genetic algorithm. The proposed method improves the performance of the cloud system by randomly initializing a chromosomes population for a given task (ti) and locating proper cloud resources to those tasks. This leads to proper cloud resource utilization and scheduling.**

Keywords: Cloud, Cluster, Fault tolerance, Genetic algorithm, Scheduling, Distributed computing.

1. INTRODUCTION

In the area such as astronomy, bioinformatics and physics clouds are becoming an important platform [1],[2],[3],[4]. As many research institutes have deployed their operational applications on cloud. For a diverse tasks, low cost entry and composite resource provision cloud computing is an attractive platform [1]. For real-time tasks fault tolerance becomes a common challenge for cloud. A high resources failure probability due to the increased functionality and complexity of the large systems incurred by the large cloud data centers [1],[2].Distributed computing paradigm can offer an efficient solution in virtualized cloud. In virtualized cloud delivering fault-tolerant capability, especially for real time scientific workflows is crucial Managing authenticity and efficiency of resources becomes crucial issues in real time system when multiple nodes situated in clouds [3]. In financial transaction, scientific computing in the many various field clouds makes a success. Just because of that Reliability and Availability are most important between cloud provider and user. Fault tolerance and resource allocation policies for cloud environments are the main thing takes into consideration [5]. Fault-tolerant Elastic Scheduling Algorithm, FASTER, ICFWS, C-HEFT and Dynamic Fault Tolerant Scheduling Algorithm these different fault tolerant and scheduling strategies are introduced in the papers. These algorithm gives better outcome but some limitations such as the algorithm can't work when multiple host failure on cloud. Some are not robust scheduling algorithm. These limitations are overcome by using genetic algorithm for scheduling cloud tasks and proper utilization of cloud resources. This archives a proper fault tolerance scheduling.

2. BACKGROUND

To achieving fault tolerance by allocating multiple copies of tasks on different computing instances scheduling is an efficient approach. For real-time tasks in virtualized clouds Faulttolerant Elastic Scheduling Algorithms (FESTAL) is designed for efficient fault tolerant scheduling [1].For fault-tolerant scheduling problem in virtualized clouds Fault-tolerant Scheduling Algorithms for real-time scientific workflows (FASTER) is designed. The First Come First Serve strategy is adopted by this architecture. FASTER realized an extended primary-backup model that integrates the virtualization and elasticity in virtualized cloud [2].

Fault-tolerant workflow scheduling algorithm for cloud systems by combining aforementioned two simple strategies together to play their respective advantages for fault tolerance while trying to meet the soft deadline of workflow [3].Cluster based Heterogeneous Earliest Finish Time (C-HEFT) algorithm, for scientific workflow in highly distributed cloud to enhance the scheduling and fault tolerance mechanism This algorithm uses idle-time of the provisioned resources environments, to mitigate the failure of clustered tasks [4]. A dynamic resource allocating mechanism with fault tolerance is defined to improve resource utilization. In cloud computing it incorporate a backup overlapping mechanism and efficient VM migration strategy for designing novel Dynamic Fault Tolerant Scheduling Mechanism for Real Time Tasks [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 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

In real-time tasks Fault tolerance becomes a common challenge for cloud. There are number of algorithms is comprehensively addresses the issue of reliability, elasticity and schedulability of virtualized clouds. Ji Wang et.al.(2014)[1] proposed Fault-Tolerant Elastic Scheduling Algorithm (FESTAL)for Real-Time Tasksin Virtualized Clouds. The star topology communication model architecture adopts where the scheduler is responsible for scheduling the tasks to the hosts, and monitoring the status of every host. Xiaomin Zhu et.al. (2016)[2] Proposed a Fault-tolerant Scheduling Algorithms for real-time scientific workflows (FASTER) in virtualized clouds is designed for fault-tolerant scheduling problem. This architecture adopts the First Come First Serve strategy. FASTER realized an extended primary backup model that integrates the virtualization and elasticity in virtualized cloud. Guangshun Yao et.al.(2016)[3] proposed Fault-tolerant workflow scheduling (ICFWS) algorithm by combining the aforementioned two strategies together to play their respective advantages for fault tolerance while trying to meet the soft deadline of workflow.

Vinay K et.al.(2017)[4] worked on fault tolerance resubmission and task replication mechanism and designed a Cluster based Heterogeneous Earliest Finish Time (C-HEFT) algorithm to enhance the scheduling and fault tolerance mechanism for scientific workflow in highly distributed cloud. J.Soniya et.al.(2016)[5] proposed Dynamic Fault Tolerant Scheduling Algorithm (DFTS) is designed for fault tolerance mechanism and dynamic resource allocation mechanism. It improve resource utilization on cloud systems.

4. EXISTING METHODOLOGY

4.1 Fault-tolerant Elastic Scheduling Algorithm

To incorporate the features of clouds fault-tolerant mechanism that extends the conventional PB model. While supporting fault tolerance in cloud to optimize resource utilization it proposed an elastic resource provisioning mechanism. The hosts and monitoring the status of all the hosts star topology communication model is adopted by this architecture, where the scheduler is responsible for scheduling all the tasks. When a task arrives its backup copy is produced by the backup copy controller. Then, the backup copy controller delivers the two copies of the task to the real-time controller that is responsible for determining whether the two copies can be finished before its deadline. If no schedule can be found to satisfy the tasks timing constrain although new resources have been added, the task will be rejected. In addition, the resource controller monitors the status of resources. When the system is in light workload, the resource controller decides whether some VMs should be cancelled to improve the resource utilization, where some VMs keep idle for a long time. The status of backup scheduled on vkl, denoted by st(tBi), is adaptively determined by the following expression.

$$st(t_i^B) = \begin{cases} passive & \text{if } f_i^P + e_{kl}(t_i) \le d_i, \\ active & \text{if } f_i^P + e_{kl}(t_i) > d_i. \end{cases}$$

4.2 Faster Algorithm

To high efficient real-time scientific workflows a FASTER is designed, i.e., fault-tolerant scheduling algorithm for real-time scientific workflows. First come first serve policy is being adopted by this architecture. For real-time workflow fault tolerant model the traditional PB (Primary backup) model by incorporating the cloud characteristics. Resource elastic provisioning mechanism that enabling full use of the idle resource by backward shifting scheduling method, allowing fast resource provisioning through the vertical and horizontal resource scaling, avoiding unnecessary allocation of frequent resource.

$$est_j^P = \begin{cases} \max(a_j, r_{pq}) | x_{jpq}^P = 1 & \text{if } P(t_j^P) = \emptyset, \\ \max\limits_{t_i^X \in P(t_j^P)} (f_i^X + tt_{ij}^{XP}) & \text{otherwise.} \end{cases}$$

$$est_j^B = \begin{cases} \max(a_j, r_{pq}) | x_{jpq}^B = 1 & \text{if } P(t_j^B) = \emptyset, \\ \max\limits_{t_i^X \in P(t_j^B)} (f_i^X + tt_{ij}^{XB}, s_j^P) & \text{otherwise.} \end{cases}$$

4.3 ICFWS Fault-tolerance Scheduling Algorithm

In this method deadline division is used to divide the soft deadline into multiple sub-deadlines for all tasks. Second, the Initial Scheduling is used to select the fault-tolerant strategy for each task from replication and re-submission and schedule all tasks for their first execution as well as the backup copies of the tasks with replication strategy. Third, the Online Scheduling scheme is used to select suitable VM for executing it again. Based on this the replication and re-submission can be combined together for fault tolerance and the workflow can be completed under the constrained soft deadline. The Directed Acyclic Graph (DAG) is to represent the workflow to submitted from customer. A DAG=(T,E) consists of R tasks T={t1,t2....tR}, which are interconnected to each other through data and control
flow such as

E={(ti,tj,eij,contrij) | (ti,tj) €TxT, I ≠j}.

4.4 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.

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

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

4.5 Dynamic Fault Tolerant Scheduling Algorithm

The sequential technique is adopted by this architecture (Queue) i.e. First-In-First-Out to handle the user tasks. In this scheme virtual machines (VMs) will dynamically create and execute the tasks based on the schedule. Each VM contain multiple tasks that are represented by where the tasks are independent and no preemptive. The attributes are proposed which are used in a task: arrival time, deadline, task size. In fault tolerance mechanism, each task is represented in two copy i.e., Primary task, Backup task. In scheduling mechanism, three controllers, i.e., Backup controller, Resource Controller, Real Time Controller. The resource executes within the deadline time, if not then the task will rejected. The scheduler is performed by maintaining resource and status of all hosts. It can also represent all host status information and resource adjustment information.

$$RTH = \underset{t_i\, eT, h_k \epsilon H}{max}\{TET/\hat{H}AT\}$$

5. ANALYSIS AND DISCUSSION

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

Fault Tolerant and Scheduling Techniques.	**Advantages**	**Disadvantages**
Fault tolerant Elastic Scheduling Algorithm.	Good performance. Effectively reduce the active time of hosts. It gives high performance in terms of resources utilization.	The architecture is based on star topology if the central node or hub may crash then the whole system collapse. The algorithm can't work when multiple host failure on cloud.
FASTER Algorithm	It doesn't keep resources idle also incorporates task overlapping. It gives high performance in terms of resources utilization. It gives best solution for scalability also it is cost efficient.	It cannot handle multiple host failure. It cannot take communication fault into consideration. It doesn't give prediction for schedulability
ICFWS faulttolerance scheduling Algorithm	It does not need any parameters from users and can be immediately applied in any Cloud computing platform.	Performance fluctuation of VMs plays a negative effect to the execution of workflow. It is not robust scheduling algorithm. It can't work when multiple deadline of workflow occur.
C-HEFT Algorithm	Produce efficient cluster based task scheduling	It does not provide accurate model in an unstable environment.

	and mapping of heterogeneous resources. Improve the load balancing. Handles failure during runtime.	It does not provide efficient fault prediction and workload model.
Dynamic Fault Tolerant Scheduling Algorithm	Ability of dynamic resource allocation. It helps to improve dynamic resource expansion, dynamic resource contraction and primary and backup scheduling	Can't work when multiple host failure on cloud. Does not add the security level parameter for the resources.

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

6. PROPOSED METHODOLOGY

This section provides the proposed methodology which has used a concept of heuristic processing as heuristic processing uses judgmental rules known as knowledge structures that are learned and stored in memory. It is governed by availability, accessibility, and applicability.

Static heuristic processing is considered in proposed method and a GA is used to schedule the various tasks. For the complete set of tasks is known prior to execution, a static heuristic is suitable for the situation. Static strategies are performed under two assumptions. The first is that tasks arrive simultaneously (The time when task ti arrives) ci = 0. The second is that machine available time aj is updated after each task is scheduled.

In large solution spaces GA is a heuristic to search for a near optimal solution. Randomly initializing a population of chromosomes (possible scheduling) for a given task (ti) is the first step. Each chromosome has a fitness value (make span) that results from the scheduling of tasks to machines (mj) within that chromosome. After the generation of the initial population, all chromosomes in the population are evaluated based on their fitness value, with a smaller make span being a better mapping. Selection scheme probabilistically duplicates some chromosomes and deletes others, where better mappings have a higher probability of being duplicated in the next generation. The population size is constant in all generations. Next, the crossover operation selects a random pair of chromosomes and chooses a random point in the first chromosome. The crossover exchanges machine assignments between corresponding tasks. A mutation operation is performed after the crossover. Mutation randomly selects a chromosome, then randomly selects a task within the chromosome, and randomly reassigns it to a new machine. After evaluating the new population, another iteration of GA starts, including selection, crossover, mutation, and evaluation. Only when stopping criteria are met, will the iteration stop.

Algorithm

Step 1: Initialize population with random chromosomes for given task (ti) having some make span value.

Step 2: On the basis of their make span value evaluate each chromosomes on the basis of their make span value.

Step 3: After evaluating select smaller make span Chromosomes.

Step 4: Duplicates some chromosomes and deletes others from selected.

Step 5: While termination condition is not true do

- Select individual for the next generation (In the next generation there is a higher probability of being duplicated)

- Recombine pairs of parents by selecting random chromosomes.

- Mutate the resulting offspring (randomly selects a chromosome, then randomly selects a task within the chromosome, and randomly reassigns it to a new machine.)

Step 6: Evaluate each candidate solutions

Step 7: End

Algorithm: Genetic Algorithm

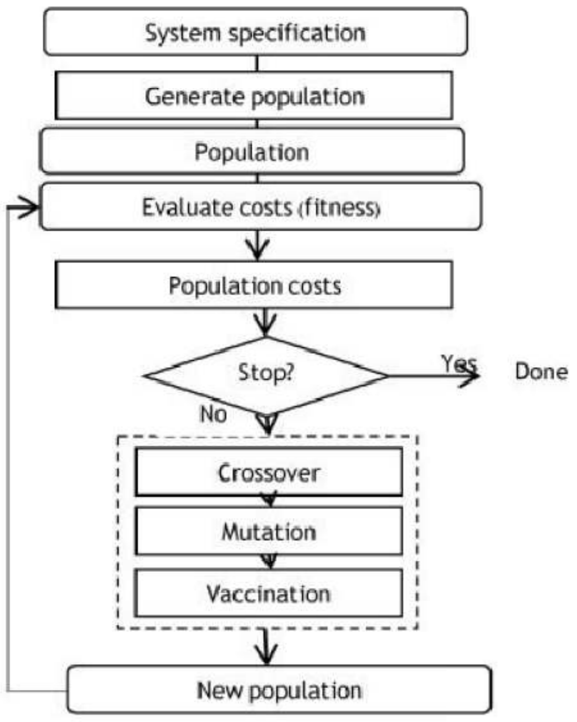

Figure 1: Flow chart for GA.

7. POSSIBLE OUTCOMES AND RESULTS

Genetic algorithms search parallel from a population of points. Therefore, it has the ability to avoid being trapped in local optimal solution like traditional methods, which search from a single point. It uses probabilistic selection rules, not deterministic ones and work on the Chromosome, which is encoded version of potential solutions' parameters, rather the parameters themselves. Genetic algorithms use fitness score without other derivative or auxiliary information, which is obtained from objective functions. Some drawbacks of proposed method are finding the optimal solution to complex high-dimensional, multimodal problems often requires very expensive fitness function evaluations. The proposed method is only considering those tasks which are already known thus operating on dynamic data sets is very difficult.

8. CONCLUSION

This paper focused on the study of different fault tolerant and scheduling techniques for cloud i.e. Fault-tolerant Elastic Scheduling Algorithm, FASTER algorithm, ICFWS fault tolerance scheduling algorithm, C-HEFT Algorithm and Dynamic Fault Tolerant Scheduling Algorithm. The proposed method improves the performance of the cloud system by randomly initializing a population of chromosomes for a given task (ti) and locating proper cloud resources to those tasks. GA starts with including selection, crossover, mutation, and evaluation then randomly selects a task within the chromosome, and randomly reassigns it to a new machine. Only when stopping criteria are met, will the iteration stop.

9. FUTURE SCOPE

The proposed method of a static heuristic is suitable for the situation where the complete set of tasks is known prior to execution. The proposed method is not useful for dynamically task arrived on system so it can't do dynamic scheduling. Future study tries to overcome this issue.

REFERENCES

1. Ji Wang, WeidongBao, Xiaomin Zhu, Member, IEEE, Laurence T. Yang, Senior Member, IEEE, and Yang Xiang, Senior Member, IEEE,"FESTAL: Fault-Tolerant Elastic Scheduling Algorithm for Real-Time Tasks in Virtualized Clouds.",IEEE Transaction on Computers,Vol. x, Issue no. x, PP. 1-14,February-2014.

2. Xiaomin Zhu, Member, IEEE, Ji Wang, HuiGuo, Member, IEEE, Dakai Zhu, Member, IEEE, Laurence T. Yang, Senior Member, IEEE, and Ling Liu, Fellow, IEEE,"Fault- Tolerant Scheduling for Real-Time Scientific Workflows with Elastic Resource Provisioning in Virtualized Clouds." IEEE Transaction on Parallel and Distributed Systems,Vol. 27, Issue no. 12, PP 3501- 3516,December-2016.

3. Guangshun Yao, Yongsheng Ding, Senior Member, IEEE, KuangrongHao,"Using imbalance characteristic for fault-tolerant workflow scheduling in Cloud systems", IEEE Transaction on Parallel and Distributed Systems, Vol. x, Issue no. x, PP 1-12, April-2016.

4. Vinay K, S M Dilip Kumar, "Fault-Tolerant Scheduling for Scientific Workflows in Cloud Environments.", IEEE-2017, Vol. x, Issue no. x, PP 150-155,July 2017.

5. J.Soniya, J.Angela, T.Revathi, "Dynamic Fault Tolerant Scheduling Mechanism for Real Time Tasks
in Cloud Computing", IEEE Transaction on
Computers, Vol. x, Issue no. x, PP 124-129,May-2016