Volume 3, Issue 7, July 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

An Efficient Fault Tolerance Mechanism Based on Moving Averages Algorithm

Amritpal Singh*
M.Tech Research Scholar
Department of Computer Science Engineering
SGGSWU, Fatehgarh Sahib, India

Supriya Kinger
Assistant Professor
Department of Computer Science Engineering
SGGSWU, Fatehgarh Sahib, India

Abstract— Fault Tolerance is concerned with all the techniques necessary to enable a system to tolerate faults remaining in the system after its development and often we encounter these failures using a well defined Fault Tolerance Mechanism (FTM).This technique is enabled using a Virtual Machine (VM) migration policy. This paper proposed a new method of identifying the need to migrate VM Machines, whenever there is an overload of work in the VM(s). The proposed algorithm provided with a way of computing the moving (dynamic) current thresholds as per current load of the data center calculated on the basis of effectiveness factor, which further provides us with implementation of a policy which remains refresh, since the payload of the datacenter varies during the day.

Keywords— Cloud Computing, Fault Tolerance, VM Migration, Moving Averages, Effectiveness Factor

I. Introduction

Cloud computing is an Internet-based network made up of large numbers of servers - mostly based on open standards, and inexpensive [1]. It uses the internet and central remote servers to maintain data and applications and provides the facility to access shared resources and common infrastructure, offering services on demand over the network to perform operations that meet changing business needs [2, 3]. Fault Tolerance is concerned with all the techniques necessary to enable a system to tolerate software faults remaining in the system after its development. Failures are likely to be more frequent in systems with thousands of processor. High-performance systems with thousands of processors have been introduced in the recent past, and the current trends indicate that systems with hundreds of thousands of processors should become available in the next few years. In systems of this scale, reliability becomes a major concern, because the overall system reliability decreases with a growing number of system components. Hence, large systems are more likely to incur a failure during execution of a long-running application [4]. The aim of the present study is to provide a solution to the problem of overloading or load balancing in cloud computing. The present study attempts to provide a model of machine migration based upon random choice policy and hence ensure optimum machine utilization with no compromise on Quality of Service. The proposed work is based on Moving Averages Algorithm. The remaining of the paper is organized as follows. In Section 2 we discuss the basic terminologies. In Sections 3 and 4 we presented related work and proposed work. In Section 5 we presented the obtained experimental results. We discuss future research directions and conclude the paper in Section 6.

II. Basic Terminologies

A. Fault Tolerance in Clouds

Fault tolerance is concerned with all the techniques necessary to enable a system to tolerate software faults remaining in the system after its development. Therefore, schemes for dealing with faults become increasingly important. The services can be either storage service or computation service. These services can be configured dynamically by making use of virtualization [4, 5]. Any application in cloud computing environment can be represented by a workflow. However, this computing environment still cannot deliver the quality, robustness and reliability that are needed for the execution of various workflows because of different failures like link failure, failure of server providing the service, malicious code in the executing node, datasets required by the task may be locked by other tasks etc. The scheduling system in a cloud should overcome such failures which can be provided by fault tolerant scheduling algorithms [5].

B. Fault Tolerance Types

There are two major faults tolerance policies in clouds:

1) Reactive Fault Tolerance

Reactive fault tolerance policies reduce the effect of failures on application execution when the failure effectively occurs. There are various techniques which are based on these policies like checkpoint/Restart, Replay and Retry and so on [5].

a) *Check pointing/ Restart*: When a task fails, it is allowed to be restarted from the recently checked pointed state rather than from the beginning. It is an efficient task level fault tolerance technique for long running applications.

b) *Replication*: Various task replicas are run on different resources, for the execution to succeed till the entire replicated task is not crashed. It can be implemented using tools like HAProxy, Hadoop and AmazonEc2 etc.

c) *Job Migration:* During failure of any task, it can be migrated to another machine. This technique can be implemented by using HAProxy.

2) Proactive Fault Tolerance

It refers to avoiding failures, errors and faults by predicting them in advance. Some of the techniques which are based on these policies are Preemptive migration, Software Rejuvenation etc.

a) *Proactive Fault Tolerance using Self Healing*: When multiple instances of an application are running on multiple virtual machines, it automatically handles failure of application instances.

b) *Proactive Fault Tolerance using Preemptive Migration:* Preemptive Migration relies on a feedback-loop control mechanism where application is constantly monitored and analyzed [5].

C. Virtual Machine Management

1) Virtual Machine Migration for Fault Tolerance

Some vendors have implemented VM migration in their virtualization solution—a big advantage for application uptime in a data center. What is VM migration? Consider the case of a server with a hypervisor and several VMs, each running an OS and applications. If you need to bring down the server for maintenance (say, adding more memory to the server), you have to shut down the software components and restart them after the maintenance window—significantly affecting application availability. VM migration allows you to move an entire VM (with its contained operating system and applications) from one machine to another and continue operation of the VM on the second machine. You can perform this migration after suspending the VM on the source machine, moving its attendant information to the target machine and starting it on the target machine [6]. To lower the downtime, you can perform this migration while the VM is running (hence the name "live migration") and resuming its operation on the target machine after all the state is migrated. The following are the benefits of virtualization in a cloud-computing:

a) *Elasticity and scalability:* Firing up and shutting down VMs involves less effort as opposed to bringing servers up or down.

b) *Workload migration:* Through facilities such as live VM migration, you can carry out workload migration with much less effort as compared to workload migration across physical servers at different locations [6].

2) Dynamic VM Consolidation

We split the problem of dynamic VM consolidation into four parts:

a) Determining when a host is considered as being overloaded requiring migration of one or more VMs from this host.

b) Determining when a host is considered as being under loaded leading to a decision to migrate all VMs from this host and switch the host to the sleep mode.

c) Selection of VMs that should be migrated from an overloaded host [7].

d) Finding a new placement of the VMs selected for migration from the overloaded and under loaded hosts.

III. **Related Work**

Malik et al. [8] have proposed a fault tolerance model for real time cloud computing. In the proposed model, the system tolerates the faults and makes the decision on the basis of reliability of the processing nodes, i.e. virtual machines. The proposed technique is based on the execution of design diverse variants on multiple virtual machines, and assigning reliability to the results produced by variants. The virtual machine instances can be of same type or of different types. The system provides both the forward and backward recovery mechanism, but main focus is on forward recovery. The main essence of the proposed technique is the adaptive behavior of the reliability weights assigned to each processing node and adding and removing of nodes on the basis of reliability. Arockiam et al. [9] have formulated a technique so as to achieve optimum fault tolerance to the virtual machines. In this paper, a middle layer is proposed and it can be placed between application layer and virtualization layer in cloud system architecture. The Purpose of this middle layer is to tolerate node failure. This layer can be seen as an assemblage of various components, each with a specific functionality and it makes use of combinations of various fault tolerant strategies to achieve optimum result. The Performance of this middle layer is user transparent too, i.e., considering economic factors, dependability factors and user's interest, it makes use of different permutations. Tchana et al. [10] have analysed the implementation of fault tolerance in a complex cloud computing environment with a focus on autonomic repair. They have shown that in most of current approaches, fault tolerance is exclusively handled by the provider or the customer which leads to partial or inefficient solutions, while collaborative solutions are much promising. They have illustrated this discussion with experiments where exclusive and collaborative fault tolerance solutions are implemented in an autonomic cloud infrastructure that they prototyped. Beloglazov et al. [11] have proposed a method for dynamic consolidation of VMs based on adaptive utilization thresholds, which ensures a high level of reaching the SLA (Service Level Agreements).They also validate the high efficiency of the proposed technique across different kinds of workloads using workload traces from more than a thousand Planet Lab servers. Dynamic consolidation of virtual machines (VMs) and switching idle nodes off allow Cloud providers to optimize resource usage and reduce energy consumption. Elmroth et al. [12] have formulated technology neutral interfaces and architectural additions for handling placement, migration, and monitoring of VMs in federated cloud environments, the latter as an extension of current monitoring architectures used in grid computing. The interfaces presented adhere to the general requirements of scalability, efficiency, and security in addition to specific requirements related to the particular issues of interoperability and business relationships between competing cloud computing infrastructure providers. In addition, they may be used equally well locally and remotely, creating a layer of abstraction that simplifies management of virtualized service components.

IV. **Proposed Work**

A. Moving Averages

Conventionally the concept of moving averages has been used in business process forecasting to determine the demand of a product based upon the latest trend. When a trend is to be determined by the moving averages, the average value for the given time period is secured and this average is taken as the average or trend value for the unit of time falling at the middle of the period covered in the calculation of the average. The effect of average is to give a smoother curve, lessening the influence of fluctuations that pull the figures away from the general trend. While using this method it is desirable to have time interval for which moving average is to be calculated in odd numbers. Since the moving average method is applied to data characterized by cyclical movements, it is necessary to select a period for moving average which coincides with the length of the cycle , otherwise cycle will not be entirely removed. This danger is more severe shorter the time period represented by the average. When the time period of moving average and the period of cycle do not coincide the moving average will display a cycle which has the same period as cycle in the data, but which has less amplitude than the cycle in the data.

$$\text{Moving Average}= \frac{a+b+c}{3}, \frac{b+c+d}{3}, \frac{d+e+f}{3} \dots\dots\dots$$

The method of moving average has several advantages like:

a) It is simple as compared to other methods.

b) It is a flexible method of measuring variations.

c) Cyclical fluctuations are automatically removed from the data.

d) The moving average has the advantage that it follows the general movements of the data and that its shape is determined by the data rather than the statistician’s choice of a mathematical function.

Present study proposes a model for systematic migration of overloaded machines to less loaded machines so that committed QoS is consistently maintained. Service Level Agreements (SLAs) with different customers may entail commitments of different levels of availability. Some SLAs may require 100% availability others may demand 99% and still others may opt for 98% availability. The machine migration system should be capable of migrating the overloaded machines as per SLA requirements. As such it is advisable to categorize the machines as per the load. The study proposes “effectiveness factor” for categorization, Effectiveness Factor (EF) may be described as:

$$EF = \frac{No.of\ error\ free\ jobs\ completed}{Total\ no.of\ jobs\ processed}$$

The “Effectiveness Factor” is calculated on hourly basis and 3 hourly moving average of EF is used for classifying the machines.

Category – 1 (100% Availability requirements): If moving average for a machine for “Effectiveness Factor” is 1, it is included in category 1 and work pertaining to SLAs requiring 100% availability is migrated to these machines.

Category – 2 (99% Availability requirements): If moving average for a machine for “Effectiveness Factor” is .99, it is included in category 1 and work pertaining to SLAs requiring 99% availability is to be migrated to these machines.

Category – 3 (98% Availability requirements): If moving average for a machine for “Effectiveness Factor” is .98, it is included in category 1 and work pertaining to SLAs requiring 98% availability is to be migrated to these machines.

It may be noted that this classification is not static but dynamic, such that if at any given time a machine is categorized into Category 1 and at another time it may be categorized into category 2 or 3 based upon value of 3 hourly moving average of EF.

For the purpose of this study it is assumed that “Effectiveness Factor (EF)” is inversely proportional to work load on the machine i.e. if the work load on the machine is high EF will be low, indicating drop in effectiveness of the machine. Categorization of machines using moving average method represents more rational approach to load migration (Fig. 1) as compared to random policy migration.

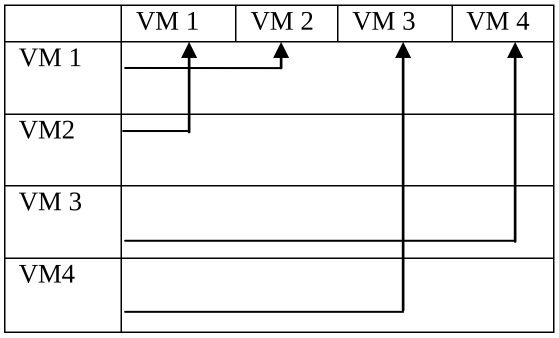

Fig. 1 Moving Average Method for Machine Migration (only selected migrations are allowed)

V. EXPERIMENTAL RESULTS

This chapter discusses and summarizes the results of the study. As already stated we have developed a model of machine migration based upon moving averages. The model approaches machine migration in a systematic manner.

A. *Designing Machine Migration Moving Average Simulator (MMMAS)*

A machine migration moving average simulator was developed and data so obtained were used to decide machine migration. Table 1. shows MMMAS.

TABLE 1. Machine Migration Moving Average Simulation Values

Time	**MA (Day 1)**	**MA (Day 2)**	**MA (Day 3)**	**MA (Day 4)**	**MA (Day 5)**	**MA (Day 6)**	**MA (Day 7)**
12:00 - 1:00 AM		0.98	0.99	0.99	1	0.98	1
1:00 - 2:00 AM	0.99	0.99	0.99	0.99	0.99	0.99	0.99
2:00 - 3:00 AM	0.98	0.99	0.99	0.99	0.98	0.99	0.98
3:00 - 4:00 AM	0.98	0.99	0.99	0.99	0.98	0.99	0.98
4:00 - 5:00 AM	0.98	0.99	0.99	0.99	0.98	0.99	0.98
5:00 - 6:00 AM	0.99	0.99	0.99	0.98	0.99	0.98	0.99
6:00 - 7:00 AM	0.99	0.99	1	0.99	0.99	0.99	0.99
7:00 - 8:00 AM	1	1	1	0.99	1	0.99	1
8:00 - 9:00 AM	0.99	1	1	0.99	0.99	0.99	0.99
9:00 - 10:00 AM	0.98	0.99	1	0.99	0.98	0.99	0.98
10:00 - 11:00 AM	0.98	0.99	1	0.99	0.98	0.99	0.98
11:00 - 12:00	0.98	0.98	0.99	1	0.98	1	0.98
12:00 - 1:00 PM	0.99	0.99	0.99	1	0.99	1	0.99
1:00 - 2:00 PM	0.99	0.99	0.99	0.99	0.99	0.99	0.99
2:00 - 3:00 PM	0.99	1	1	0.99	0.99	0.99	0.99
3:00 - 4:00 PM	0.98	0.99	0.99	0.99	0.98	0.99	0.98
4:00 - 5:00 PM	0.97	0.99	0.98	0.99	0.97	0.99	0.97
5:00 - 6:00 PM	0.98	0.99	0.98	0.99	0.98	0.99	0.98
6:00 - 7:00 PM	0.99	0.99	0.99	0.99	0.99	0.99	0.99
7:00 - 8:00 PM	1	1	1	1	1	1	1
8:00 - 9:00 PM	1	1	1	0.99	1	0.99	1
9:00 - 10:00 PM	0.99	1	0.99	0.99	0.99	0.99	0.99
10:00 - 11:00 PM	0.99	0.99	0.99	0.99	0.99	0.99	0.99
11:00 - 12:00	0.99	0.99	0.99	1	0.98	1	

Table 1. represents simulated 3 hours moving averages for a machine. As can be seen from the table this machine can be classified into Category 1, 2 or 3 depending upon the moving average value of the Effectiveness Factor (EF) of the machine at a particular time. Moving averages tend to smoothen the variations over a period of time and hence are likely to provide more realistic approximation for machine migration. A more comprehensive understanding of the above table can be made through individual day wise graphs. Fig. 2 shows moving average for the day 1.

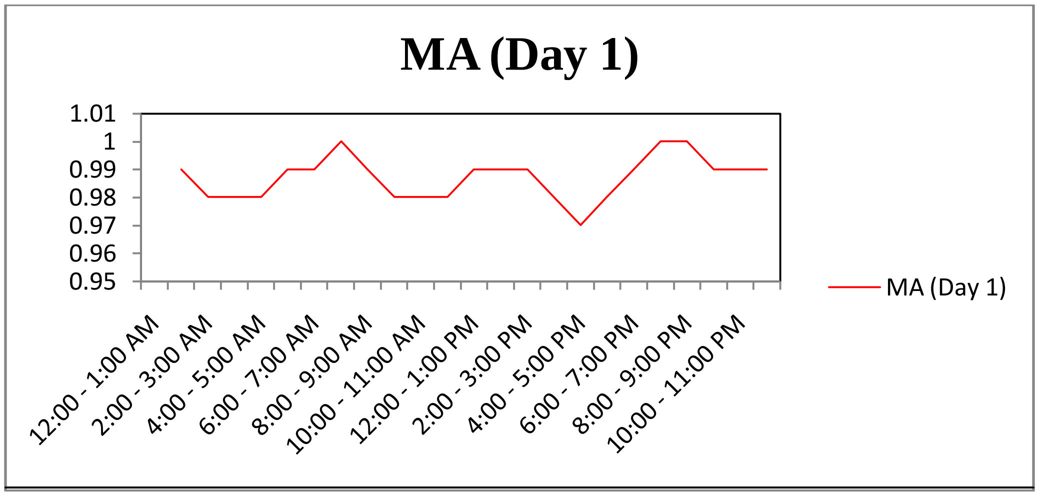

Fig. 2 3 Hourly Moving Average of Effectiveness Factor (Day 1)

Following inferences can be drawn from the graph:

1) Between 2:00 AM to 5:00 AM machine has Moving Avg. EF of 0.98, indicating that machine is overloaded and it cannot be further loaded with task which entails 100% availability, but can be loaded with the task requiring 98% availability. Thus during this period machine falls into Category 3.

2) Between 5:00 AM to 7:00AM moving average improves to 0.99 indicating a slighter lesser load as compared to earlier time period and now it falls into category 2. This suggests that now this machine can be loaded with SLAs requiring 99% availability. However, it still cannot be used for SLAs requiring 100% availability.

3) Between 7:00 AM to 8:00 AM moving averages further shows improvement and reaches 1.00, indicating now the machine can be trusted with work requiring 100% availability and falls into category 1.

4) Between 04:00 PM to 05:00 PM moving average is 0.97, indicating that at this particular time machine is heavily loaded and some work needs to be shifted to other machine to make it usable for SLAs requiring 98% availability.

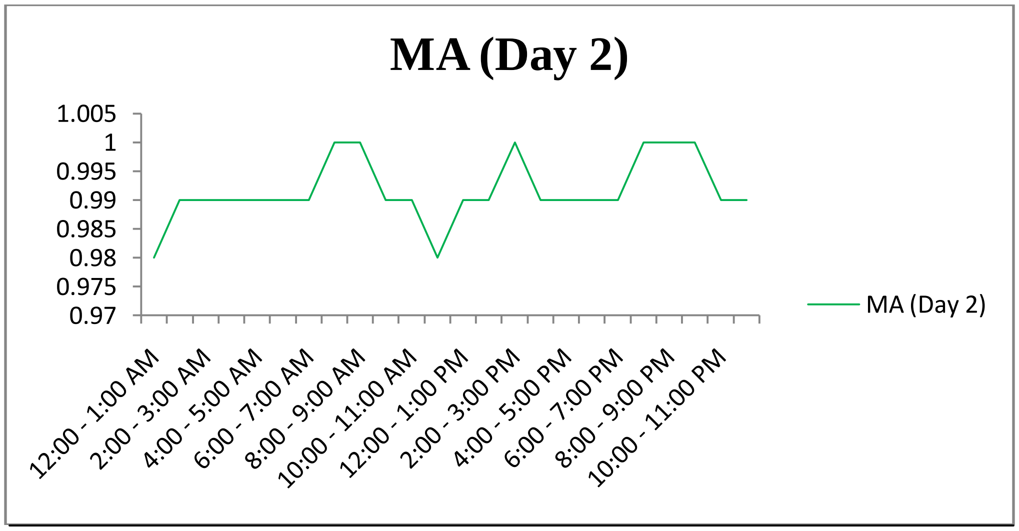

Fig. 3 3 Hourly Moving Average of Effectiveness Factor (Day 2)

A comparison of fig. 2 with fig. 3 (showing moving average for the same machine on day 2) exhibits some interesting facts as below:

1) In fig. 3 moving average between 1:00 AM to 07:00 AM is 0.99, which is significantly at variation with respect to moving average of the preceding day (0.98). Hence on day 2 for this particular time slot the machine is classified into category 2 (while on day 1 it was category 3).
2) EF moving average falls to 0.98 only once during the 24 hour period between 11:00 AM to 12:00 AM. Otherwise throughout the 24 hour period MA is between 0.99 to 1.00.
3) While moving average between 04:00 PM to 05:00 PM was 0.97 on day 1, while on day 2 it is at 0.99 .

VI. CONCLUSION AND SCOPE OF FUTURE WORK

In this current research work we have concluded that the proposed algorithm provided us with a way of computing the moving (dynamic) current thresholds as per current load of the data center calculated on the basis of effectiveness factor, which further provides us with implementation of a policy which remains refresh, since the payload of the datacenter varies during the day. It provides a model with works on latest figures obtained on assumption that the data to be processed by the datacenter maintains a normal distribution curve, however it may not be the case, sometimes, it may happen that the data traffic pattern may take a exponential distribution curve, therefore for future scope we suggest that the work must be done on this basis to calculate the Effectiveness Factor (EF).

ACKNOWLEDGMENT

I am thankful to my guide Supriya Kinger, Assistant Professor in Department of Computer Science Engineering for their valuable support. Also I am thankful to my Department for the technical support.

REFERENCES

[1] J. Udayakumar, M. Manikkam and A. Arun, "CLOUD-SLA: Service Level Agreement for Cloud Computing," International Journal of Computing Science and Information Technology, Volume 1 (02), pp. 13-16, 2013.
[2] Q. Zhang, L. Cheng and R. Boutaba, " Cloud Computing: state-of-the-art and research challenges," Journal of Internet and Services and Applications, Volume 1,Issue 1, pp. 7-18, 2010.
[3] S. Rao, N. Rao and E. Kumari, "Cloud Computing: An Overview," Journal of Theoritical and Applied Information Technology, Volume 9 (01), pp. 71-76, 2012.
[4] S. Chakravorty, C. Mendes and L. Kale, "Proactive Fault Tolerance in MPI Applications via Task Migration," IEEE International Conference on High Performance Computing (HiPC), pp. 1-12, 2006.
[5] A. Bala and I. Chana, "Fault Tolerance-Challenges,Techniques and Implementation in Cloud Computing," International Journal of Computer Science Issues, Vol. 9 Issue 1, No 1, pp. 288-293, 2012.
[6] M. Ahmed, A. Chowdhary,M. Ahmed and M. Rafee, "An Advanced Survey on Cloud Computing and State-of-the-art Research Isuues," International Journal of Computer Science Issues,Vol. 9, Issue 1, No 1, pp. 201-207, 2012.
[7] A. Beloglazov and R. Buyya, "Optimal Online Deterministic Algorithims and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centres," Wiley InterScience, pp. 1-24, 2012.
[8] S. Malik and F. Huet, "Adaptive Fault Tolerance in Real Time Cloud Computing," IEEE World Congress on Services, pp. 280-287, 2011.
[9] L. Arockiam and G. Francis, "FTM-A Middle Layer Architecture for Fault Tolerance in Cloud Computing," International Journal of Computer Applications, pp. 12-16, 2012.
[10] A. Tchana and L. Broto and D. Hagimont, "Approaches to Cloud Computing Fault Tolerance," Computer, Information and Telecommunication Systems (CITS), pp. 1-6, 2012.
[11] A. Beloglazov and R. Buyya, "Adaptive Threshold-Based Approach for Energy-Efficient Consolidation of Virtual Machines in Cloud Data Centres," 8th International Workshop on Middleware for Grids, Clouds and e-Science, pp. 1-6, 2010.
[12] E. Elmroth and L. Larson, "Interfaces for Placement,Migration, and Monitoring of Virtual Machines in Federated Clouds," 8th International Conference on Grid and Cooperative Computing, pp. 253-260, 2009.