*IOSR Journal of Computer Engineering (IOSR-JCE)*
*e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 4, Ver. I (Jul - Aug 2017), PP 64-73*
*www.iosrjournals.org*

# An Adaptive Fault Reduction Scheme to Provide Reliable Cloud Computing Environment

*M. Damodhar[1], S. Poojitha[2]
[1](Computer Science and Engineering, Sri Venkateswara University College of Engineering, India)
[2](Computer Science and Engineering, Sri Venkateswara University College of Engineering, India)

***Abstract:*** *The Now a days Cloud technologies has become an essential trend in the market of information technology. Virtualization and Internet-based Cloud computing leads to different kinds of failures to arise and hence, leads to the requirement for reliability and availability has become an essential issue. To make ensure for reliability and availability of cloud technologies, a scheme for fault tolerance need to be developed and implemented. As the majority of the early schemes for fault tolerance were focused on the utilization of only one way to tolerate faults. This paper presents an adaptive scheme that deals with the difficulty of fault tolerance in various cloud computing environments. The projected scheme employs adaptive behavior during the selection of replication and fine-grained checkpointing methods for attaining a reliable cloud platform that can handle different client requests. In addition to this the algorithm also determines the best suited fault tolerance method to every designated virtual machine. As the proposed scheme adapts fault tolerance methods it improves the performance of the cloud in terms of availability, throughput, cloud overheads due to huge number of check points and maintenance cost.*

***Keywords:*** *Adaptive fine-grained checkpointing(AFC), Cloud technologies, Fault tolerance, replication, virtual machines.*

## I. Introduction

The present market of Information Technology has a significant change due to the existence of cloud technologies, which become an essential part of online businesses [1]. Now-a-days, most of the businesses, from small to big enterprises, migrated to cloud computing to attain high productivity level by delivering their issues regarding IT to experts. Cloud computing provides a wide-ranging of IT services and solutions for both companies and individual users [2]. Users or companies can lease cloud components and their services without spending time as well as money in building or buying these components. In cloud systems, computing is hosted as an abstract service through Internet with hiding the details of their implementation.

Besides the rapid growth of computing and communication techniques, a vast deal of data is generated. These huge data requires more robust computation resources and extreme storage space. Over the past years, cloud computing fulfills the requirements of application and develops more quickly. Basically, it takes the data processing as a service, like storage, data security, computing, etc. With the utilization of public cloud platform, the clients were relieved from the problem of storage management, universal access of data along with independent geographical locations, etc. Hence, more number of clients would prefer to store and process their data with the help of remote cloud computing system.

The deployment models of cloud computing systems are public, private or hybrid. In public, services are provided through the Internet in forms of cloud practical application. The main categories of these applications include Infrastructure-as-a-Service (IaaS), Software-as-a-Service (SaaS) and Platform-as-a-Service (PaaS). Most of IT businesses cannot invest in these services such as supercomputer class services. In IaaS, the cloud provides computing, storage and networking resources with any required configuration and capacity as paid services to the customers. Examples of practical applications of IaaS can include Amazon EC2 and Google Compute Engine. In most IT organizations, there are no enough experts to develop and run the required software applications. In SaaS, the cloud pro- vides customers with access to professionally implemented software applications and thus they save the customers' money. Salesforce.com and Google Apps are examples of practical applications of SaaS. In PaaS, customers can run their custom applications on the general purpose software and hardware with the most recent configurations. Practical applications of PaaS include Google App Engine and Microsoft Azure [3].

Private clouds are implemented and maintained by enterprises to deliver internal services also they have more flexibility when compared with public clouds but they are more expensive. In hybrid clouds known as combined version of public and private clouds, some portions of computing will be performed in a public cloud while the remaining portions will be performed internally through the private one.

In spite of cloud computing systems that are used to offer services of computing, they are not perfectly reliable and they could suffer from outages of services due to failures [4]. An outage is well-defined as the case

in which the request of a customer is not completed in its specified deadline. With the rise of cloud users, the number of required services increases as a result the probability of outages increases. The foremost reason of these outages includes software failures such as extreme work load, incorrect upgrade, hacking, and so on. In addition to software failures hardware failures include network failures, unavailable resources, power fluctuations, etc.

Outages are general in public clouds where enormous services were delivered to customers with required quality in service. In the past decade, many outages have occurred in most famous public cloud environments. In 2013, the home page of Amazon went down for nearly an hour, which in turn leads Amazon to costs close to five million US Dollars. In 2014, a number of Google services such as Gmail, Google Docs and Calendar were stumbled for nearly an hour. Some of the Google servers receive incorrect configurations which causes a wide-ranging of errors [5]. In 2015, some services of Microsoft Azure cloud such as websites and virtual machines had more than two and half hours of interruptions in different regions.

Cloud outages or failures have a great impact on both the cloud vendors and the customers. For vendors, a profit will be lost due to the cloud resources that will be used in order to alleviate the effects of outages occurred. K. Bilal et al [6] have stated that each downtime hour in a data center costs around U$ 50,000. For customers, their requirements, such as deadline time, may not be achieved. So, there is a great need for a reliable and available cloud with a dynamic method of fault tolerance. The method should transparently remove or reduce to some extent the effects of failures on both customers and profit needs.

The methods that deal with Fault tolerances can be reactive or proactive. The key role of the reactive methods is to decrease the effect of occurring faults whereas the aim of the proactive methods is they can avoid the fault occurrences. A reactive method generally includes the combination of replication and checkpointing mechanisms. Almost all of the cloud computing systems were depended on reactive methods that especially include replication [7].

The replication method considers the probability of a particular VM failure is enormously larger than the simultaneous occurrence in the failures of multiple VMs. It permits multiple virtual machines to initiate simultaneous execution of redundant copies of the same task of the customer in order to exclude recomputation performed by it from scratch during the failure situation. Consequently, the service can be provided efficiently to users with no disturbance to their QoS requirements during failure situations. In checkpointing mechanism, the cloud intermediately saves the state of execution for both the request under execution and the executing VM to a stable storage so as to reduce the time for recovery during failure situations. In case a failure arises, rather than restarting the execution of request from its beginning, it will be started from the computation point where the last checkpoint was saved in the memory [7].

The key contribution of this paper is to present an adaptive fault reduction scheme that can cope for both proactively and reactively with the difficulty of fault tolerance in environments of cloud computing. For the case of proactive, the scheme depends on requirements of a user for specific request and the information available about VMs during the scheduling time. Furthermore, the scheme employs both fine grained checkpointing and replication methods as well as it adaptively select the best suited method that depends on the existing conditions of the cloud environment.

The subsequent part of the paper is organized as follows: Section 2 presents a brief illustration of the related work. Section 3 describes the problem. Section 4 provides the details of the proposed scheme. In Section 5, results obtained are presented and finally the paper was concluded in Section 6.

## II. Background And Related Work

### A. BACKGROUND

The dynamic behavior of the cloud increases the probability of failures. In order to reduce or escape from the problems and difficulties of these failures, the cloud needs fault tolerance that might be reactive or proactive. Reactive fault tolerance methods were applied to reduce or to omit the influence of failures on costs associated with time and monetary. Replication and checkpointing are the two methods that were generally used as reactive methods.

The replication method depends on the probability of failures will be minimized when multiple virtual machines are used to handle the same request of a particular user. Request recompilation is avoided by executing multiple replicas of the same request on different virtual machines at the same instant. If a virtual machine fails the cloud can still capable to execute the task within the boundaries of user's requirements. The consequences of the virtual machine that finishes the task first were considered and consequences of other virtual machines were ignored [8].

Checkpointing is the subsequent reactive method. In checkpointing, the user's request execution status is repeatedly saved to a stable and safe storage at the execution time. In failure situation, the cloud can continue to execute the request starting from the last point in where status was recorded. This avoids restarting the service of the request from its initial point of execution. Even though it can reduce the response time of carrying out a

request, but it results in more wasted time. It is because of the recovery of a virtual machine from its failed state if it is the only one that can accomplish the task. On the other hand, it is essential for the cloud to utilize this method if there is only a single virtual machine that can handle the user's request. Checkpoint interval is known to be the time between two checkpoints[9].

On the other hand, proactive methods are probabilistic and they are employed in order to predict to a possible extent the faults of virtual machines before their occurrence. The major goal of these techniques is trying to avoid the failure occurrences and later avoids the recovery procedures of the reactive methods. During requests scheduling, proactive methods chooses scheduling decisions based on the prior failure information about available virtual machines. Consequently, the number of upcoming failures can be minimized and the cloud reliability will be enhanced.

**B. RELATED WORK**

Fault tolerance is one of the key issues in distributed computing systems like grid and cloud computing systems. In grid computing, there is a lot of fault tolerance work have been done in the literature, whereas a slight research has been devoted cloud computing environment.

In 2010, Goiri et al. [9] proposed a checkpointing scheme where the time required for storing checkpoints were reduced. They achieved this reduction in time by saving only modifications of the read-write regions. To save checkpoints they were employed in the Distributed File System of Hadoop. In 2013, Hui et al. [10] proposed a fault tolerant method with the usage of coordinated checkpoints at virtual machine level. Their method removes the difficulty of unavailability through coordinated protocols for checkpoint recovery. In 2014, Limam and Belalem [11] defined an adaptive checkpoint method with the goal of removing unnecessary checkpoints or adding of additional checkpoints based on the existing status of the cloud components. Their method increases or decreases the intervals between checkpoints with fixed rates. In 2015, J. Cao et al. [12] have presented a uniform fault tolerance scheme based on checkpointing. Their method supports for lengthy jobs and for the jobs that have priorities.

In 2013, Ganga and Karthik [7] have proposed a method based on replication with the major intention to tolerate faults when scientific work-flow systems were used. Das and Khilar [13] proposed a method based on replication for the service time reduction and to increase the availability of system. Their method uses different software's for fault tolerance on several virtual machines. Additionally, it minimizes the likelihood of upcoming faults by not scheduling tasks to server virtual machines whose rate of success is very low. Alhosban et al. presented a scheme which depends on planning and prediction. A method of recovery is selected which can be applied during fault occurrence. The selection was based on user requirements, failure history and service weight and its critical nature. Methods that were chosen are replication and retry.

In 2012, Zheng et al. [8] have proposed an algorithm that has the ability to select a fault tolerance method for every virtual machine. All methods that were selected by the algorithm are from variations of the replication method such as multi-version and parallel. In 2015, Saranya et al. [14] presented and evaluated a method that depends on both replication and tasks resubmission. Their scheme was based on the priority assigned to each and every task depending on length of the tasks, means their deadline and the out-degree of every task. In 2015, Liu and Wei proposed a replication based mechanism for fault tolerance in mapreduce framework in where they considered the failures of both hardware and software.

The analysis of literature illustrates that most of the earlier work done are essentially based on the usage of only one fault tolerance method that is either replication or checkpointing. There is very little work done that considers the usage of these two methods together for fault tolerance in cloud computing systems. Furthermore, almost all of the present replication based work considers a fixed or static number of replicas and they perform replication for every virtual machine in the cloud, which is not economically best approach. Whereas in the case of checkpointing mechanism, most of the work that was proposed assumes fixed or fixed change of the length of the checkpointing interval at the execution of the user jobs or requests. There is very small work was performed that considers the adaptive nature in the length of checkpoint interval. Hence, there is a necessary for a scheme that considers both replication and checkpointing mechanisms and adaptively chooses the number of replica or checkpoints.

## III. Problem Description

Cloud services are delivered either as storage or computing services. Google, iCloud and Dropbox are well known examples of storage services further, Amazon EC2 and Microsoft Azure are the best examples for computing services. In order to be served by the cloud, a user is supposed to submit his service request to the appropriate cloud service provider along with the requirements that are required for his request. The provider negotiates with the user in order to determine the price and the quality of service. If the user accepts, the provider will prepare the virtual machine of cloud that can handle the request and the service will start to execute the task.

Most of the cloud resources are not mainly designed to achieve the economic objective of the cloud environment. These resources are composed into several virtual machines to perform and complete the requests of cloud users. So, it is expected that a lot of failures will arise that will prolong the expected time to carry out the requests of a user which in turn exhaust the cloud resources. For users, they will not acquire their services in the time that was generally expected. For the cloud, failures will lead to the severe damage of cloud resources which in turn leads to wastage of money. This will lead to a significant impact on the credibility, reliability and cloud reputation. Therefore, it is very essential to implement fault tolerance methods in cloud computing systems in order to omit or alleviate the failure influences on the cloud performance.

Replication of both data and applications is the method used by almost all of the present cloud computing systems. It is applied in Amazon S3 through the storage of data objects on various storage units. The iCloud provides rental based infrastructure services from Microsoft's Azure or Amazon's EC2 to satisfy the replication. In spite of that, cloud outage reports refer to the point that there is still insufficiency in reliability [15]. Applying methods of fault tolerance in clouds faces the subsequent challenges were numbered below:

1. The cloud can have only a particular copy of the virtual machine that can have capability to handle the request of the cloud user. Also, the cloud may have several virtual machines that can handle the user's request, but only one is ready to available and the remaining are busy in performing other services or in some cases they are out of service. In such cases, replication method cannot be applied.
2. The number of replicas should not be fixed or static because this will lead to a pitiable influence on the cloud. This is owing to the fact that other virtual machines will be used to handle the same service. Yet, these virtual machines can be used to handle the requests of other user services. Therefore, the cloud will lose profitable charges.
3. It is not economical to implement replication for every service or virtual machine. Replication should generally require for applying to services that are allocated to the top-most valuable virtual machines that will have an excessive impact on the cloud performance if they fail. Determining the top-most valuable virtual machines is a huge challenge.
4. In the checkpointing scheme, defining the interval length of the checkpointing is a crucial challenge. Checkpointing with fixed interval could leads to redundant checkpoints that takes cloud resources which in turn increases checkpointing latency.

In order to handle with the primary challenge, checkpointing method is considered in our scheme with replication. The proposed scheme permits the cloud to select either checkpointing or replication in order to accomplish fault tolerance. With the aim of addressing the second challenge a replication algorithm that has capability to adaptively determine the number of replicas of an application. For the third challenge determined above, the profitability attained by the cloud when using the virtual machine is involved in determining the number of replicas needed for every virtual machine. For the last challenge an algorithm that has capability to adaptively determine the length of the checkpointing interval. The algorithm assumes that the gap between the checkpointing intervals must not be fixed at the time of execution of user's task. The algorithm takes the failure probability of VMs for calculating the subsequent checkpointing intervals.

## IV. Cloud Architecture

Cloud computing environments should have the capability to obtain, perform, monitor and control the requests of the user's. In order to provide cloud services it must be reliable within the boundaries of user requirements. This section describes the proposed scheme that has the capability for enabling the cloud to be reliable. As shown in Figure 1, the architecture of the scheme assumes the cloud is comprised of three significant layers: application, virtual and physical layers from top to bottom. One function of the application layer is to permit users for cloud interaction. Similarly, it handles the task of scheduling of users' requests to the virtual machines presented in the cloud. Additionally, tolerating faults is the key responsibility of the application layer. In order to accomplish these functions, the application layers structure is further classified in to four modules:

1. *Service Inspector*: This module is liable for ensuring the achievement of the QoS requirements of user's. In this paper, the considered QoS requirements comprise monetary costs and time. A cloud user is capable to submit his request to the cloud through this module with his QoS requirements. The module asks the Database Status module for availability of suitable VMs that can have capability to handle the request of user and acquires a reply. If the reply specifies the presence of suitable VMs that can have capability to handle the request within requirements of user, the Service Inspector module will takes the request and delivers it to the subsequent module that is Scheduler. Otherwise, the request of the user will be discarded simply.
2. *Scheduler*: The key function of the Scheduler is to allocate every request to the appropriate virtual machine that has capability to execute it within the bounds of user requirements. Also, it has the responsibility to determine the charge for serving the request of user. Additionally, Scheduler has the responsibility of fault

tolerance. In order to accomplish its responsibilities, the Scheduler module needs to hold the components as follows: Ranker, Pricing, Job Scheduling and Fault Tolerance Manager (JSFTM) and Dispatcher. Figure 2 illustrates the communications between the key components of the Scheduler. The key role of the Ranker sub-module is to determine the top-most valuable VMs in the cloud as determined earlier in this paper. It accepts user's request with their QoS requirements from the Service Inspector module and contacts the Database Status module in order to acquire the information about virtual machines that can complete the user's request. According to this information, it formulates a list of VMs that can satisfy the monetary cost and time requirements of the user's request. Pricing component or sub-module defines the charge to the user need to pay for the service based on the requirement. JSFTM implements Algorithm 1 in order to choose the suitable fault tolerance method for the virtual machine allocated to every request. The algorithm chooses either checkpointing or replication depending on available information of virtual machines. Dispatcher component delivers the users requests to the designated VMs.

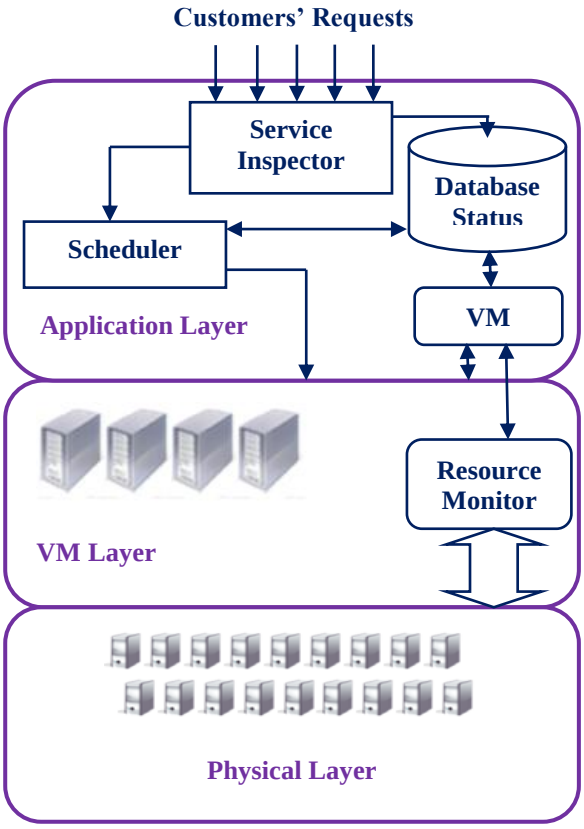



**Fig. 1.** The level architecture of a cloud computing system

3. Database Status: It represents the central information repository that has information about all virtual machines in the cloud like storage and computing capacities, price, usage history and failure history.
4. VM Monitor: The foremost function of this module is to perceive the virtual machines performance in the cloud. It frequently notifies the Database Status module to update the record of a VM during the failure situation or the recovery of that VM. Additionally, it has a job of forming or reforming virtual machines in the cloud. It contains virtualization software which is used to form unique and isolated virtual machines with the help of cloud physical resources.

**Algorithm 1** AFT Algorithm

**Input:** $C_{ij}$ is the estimated cost if request *i* is executed on $VM_j$
$T_{ij}$ is the estimated time if request *i* is executed on $VM_j$
$C_{iu}$ is the cost required by the user *u* for request *i*
$T_{iu}$ is the deadline time required by the user *u* for request *i*

i = 1;
**while** (there are requests not executed){
**for** each request *i* **do**{

```
                    Find a list of VMs that can carry out i;
                    for each VMj  in the list do{
                        if (Tij > Tiu || Cij > Ciu)       //VMj cannot execute i
                            remove VMj from the list;
                        if (list is not empty){         //The request can be executed
                            Sort the VMs list ascending based on Tij * Cij;
                            if (the list contains more than one VM)
                                Replication method is selected;
                            else
                                Checkpointing scheme is selected;
                        else {
                            Send ‘‘Request cannot be executed’’ to the QoS Controller;
                            End the algorithm for i;
                        }
                        i++;      //Subsequent request in the scheduler
                    }
                }
            }
```

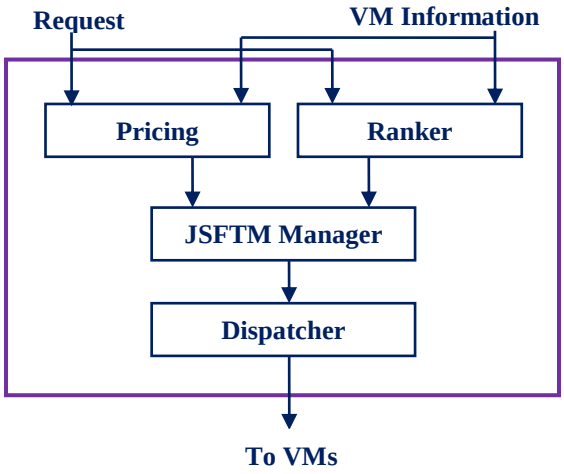



**Fig. 2.** Scheduler components and their interactions.

VM layer is the second or middle layer of the cloud that contains several virtual machines in the cloud and every virtual machine is formed with the help of one or more physical resources. Similarly, each and every physical resource can be shared and makes used by other virtual machines in the cloud. Moreover, various VMs can be emulated on a single physical resource for satisfying the requirements needed by user requests. This layer consists of a module known as Resource Monitor. The significant function of this module is to perceive the physical resources performance in the cloud and to frequently notify the VM monitor with changes or modifications happened. Changes may contain resources going to left the cloud or new resources that were joined in to the cloud. Based on these changes, Resource Monitor module can reform the virtual machines affected.

The third and the bottom layer of the cloud is the physical layer it contains both hardware and software resources of the cloud. Resources are the actual operators in the infrastructure of cloud.

### *A. AFT Algorithm*

Algorithm 1 is called the Adaptive Fault Tolerance (AFT) algorithm and it is proposed with the main goal of choosing the suitable method for fault tolerance in the cloud computing infrastructure. The algorithm is implemented in the JSFTM component of the Scheduler module. In order to attain its goal, the algorithm depends on using the requirements of a user and the virtual machines information that is available. Initially, the algorithm formulates a list of virtual machines that have capability to handle the user's request and satisfies their requirements. The user's requirements considered by the algorithm include both monetary costs and costs associated to time.

Later, the algorithm chooses fine grained checkpointing scheme if there is only a single VM in the list formulated previously or else the algorithm simply chooses replication method.

***B. Replication Algorithm***

Replication is applied when there are a number of available virtual machines in the cloud that has capability to handle the requests of a user. On the other hand, it is a crucial challenge to define the best number of replicas. Additionally, it is not an economical approach to carry out replication for every virtual machine. So, we only necessary to replicate requests executed on the top-most valuable virtual machines that will have lager impact on cloud performance if they were failed.

In order to determine the top-most valuable VMs in the cloud, virtual machines need to be ranked based on their influence and value on the cloud. The ranking is based on failure probability of the virtual machine and the profit gained through using it. Failure history of a virtual machine determines its failure probability. For every virtual machine, failure history can be rendered by the number of failure occurances, failure time, the time gap between failures and type of failures. The requirement of a virtual machine to a fault tolerance method is captured through failure probability. As the value of the failure probability increases consequently the requirement for applying fault tolerance schemes rises.

Generally, the random failure occurrences are known to be a stochastic process [16] and Jump Linear Systems (JLSs) can be used to model it as they encompass event driven and time evolving techniques. The process depends on the time period between any two consecutive faults. In cloud environment, this time period is a random variable that follows general probability distributions and the process is generally known to be semi-Markov process. The jump linear system of the semi-Markov process is known as semi-Markovian JLS with time-varying transition rates [17].

In this paper, the failure probability of a virtual machine is assumed to follow Poisson distribution which means that the number of failures in any two dissimilar or disjoint time periods is independent over the change in time.

The profit of a virtual machine is denoted as $P_i$, represents the percentage of gained cloud profit on the usage of virtual machine *i* in executing requests. The value of the virtual machine *i* presented in the cloud is determined based on its profit. The larger the profits of a virtual machine the larger the value.

The rank of a virtual machine can be computed by the scheduler modules Ranker component. The Ranker gets the failure probability values and profit of virtual machines from the Database Status module.

The fixed number of replicas is not a best choice in the environment of cloud computing because other virtual machines of same type will be used to handle the same request of a user. On the other hand, these virtual machines can be used to handle requests given by other users. As a result, charges pertaining to profits will be wasted. Furthermore, it is not economically to implement replication for every request or for every VM.

**Algorithm 2** Replication Algorithm

- $F_i(X)$ : The failure probability of a virtual machine *i*
- $P_i$ : The percentage of cloud profit gained through the usage of virtual machine *i*
- *Rep* : The number of replicas available i.e., *Rep* (*l*)(*w*) , $l$ = 0, 1, 2, … n; *and* $w$ = 0, 1, 2, … $m$; are integers
- $F_i(X)(k)$, $k$ = 0,1,2,3,….,n are integers *s.t.* 0 <= $F_i(X)(k)$ <= 1.0 and $F_i(X)(0)$ < $F_i(X)(1)$ < …. < $F_i(X)(n)$.
- $P_i$(y), y = 0,1,2,….,m are the percentage of profit gained by virtual machine i s.t. 0 <= $P_i$(y) <= 100 and

$P_i(0) < P_i(1) < P_i(2) <….< P_i(m)$

```
for (j = 0; j < n; j++){
        for (k = 0; k < m; k++){
                if (Fi(X)(j) <=Fi(X) < Fi(X)(j + 1) and Pi(k)<= Pi < Pi(k+1)
                        Rep = Rep(j)(k);
        }
}
```

Algorithm 2 proposed in this paper is the replication algorithm in order to adaptively determine the number of replicas for a request. The number of replicas will not be fixed for every request or virtual machine. In order to adaptively decide the number of replicas, the operation of the algorithm is based on both the probability of failures and the percentage of gain in the profit of a cloud by the virtual machine designated to handle the user's request. As either the probability of failure or percentage in the profit of a virtual machine goes higher the requirement for additional replicas increases. As a result, virtual machine with larger values in profit or probability of failures contains higher fault tolerance requires and then larger priority of replication than other VMs.

***C. Fine-grained Checkpoint Algorithm***

Distributed systems, such as grid computing systems were extensively uses checkpointing as a reactive fault tolerance scheme to alleviate the failure impacts when arises. Furthermore, the majority of cloud computing systems were implemented with replication techniques. Though, from the side of the service provider of cloud, replication consequences in profit loss due to allocation of additional components to handle the request replicas, mainly these components may be useful for other user's requests. In addition, from the user's perspective, replication leads to loss in time due to waiting for components that handle replicas to be free from executing other user's requests. Hence, the major advantage of using checkpointing scheme over replication is to protect the computing resources of the cloud to other users' requests and to decrease the profit losses because of using the replication methods.

Checkpointing interval and latency are the two key parameters that were largely influence a checkpointing algorithm. The checkpointing interval signifies the time between a checkpoint and the subsequent checkpoint. Checkpointing latency is the time utilized in saving a checkpoint in the storage area. In the case of very small checkpoint interval then there will be huge number of checkpoints. This huge number of checkpoints will largely utilize cloud resources while saving checkpoints and hence, large checkpointing latency is caused. Furthermore, large checkpoint interval leads to very less number of checkpoints further a significant part of the request need to be recomputed in the failure situation. This less number of checkpoints will faintly utilize cloud resources while saving checkpoints and consequently results in low checkpointing latency.

Subsequently, determining checkpointing interval length is the crucial challenge for a checkpointing mechanism. Fixed interval leads to redundant checkpoints that can make utilization of cloud resources and in turn increases in checkpointing latency. So, the key goal of this paper is to develop an algorithm that has capability to adaptively determine the checkpointing interval length based on failure history of VM. Our fine grained Checkpointing algorithm assumes that the checkpointing interval length must not be fixed during the execution of the user's requested job. The algorithm calculates the subsequent checkpointing interval at the time of the present checkpoint. It is calculated by considering the VMs failure history on which the task is going to execute. In the situation of a poor failure history, the algorithm will lessen the intervals between checkpoints. Additionally, the algorithm will lengthen the checkpoint interval if there good failure history for a VM.

**Algorithm 3** Checkpointing and Recovery Algorithm
**Input:** Customers requested task, bid and VM's
**Output:** Total task execution time and total cost

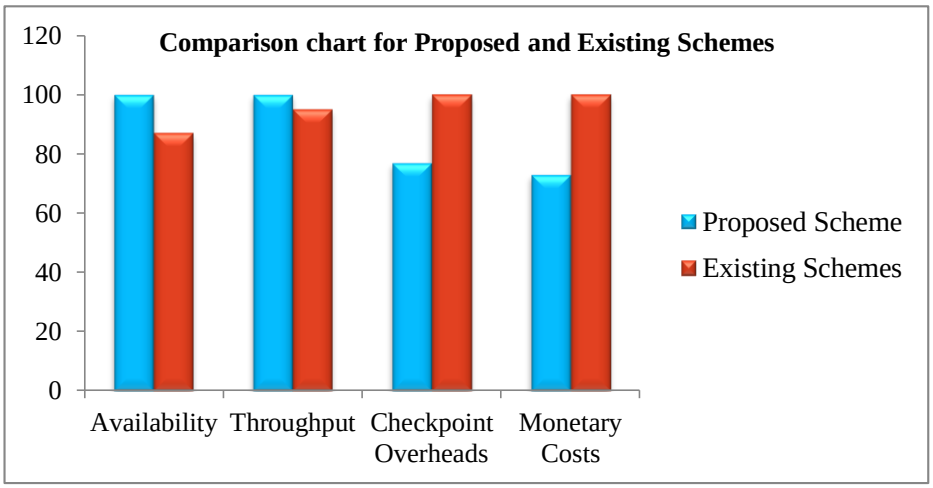
```
Boolean TF_flag = false;//Occurrence of task failure
Boolean TS_flag = true;//Represents task starting
while(!Task execution completed) do
        if(TS_flag)then
                Estimation();
                TS_flag = false;
        end if
        if(spot_prices <= Customer_bid)
                if(TF_falg)then
                        Recovery();
                        TF_flag = false;
                end if
                if(rising_edge &&Price_threshold <= spot_prices) then
                        Checkpoint();
                end if
        end if
        if(failure is occurred) then
                TF_flag = true;
                update the VM failure information.
        end if
end while
Estimation( ){
        calculate the points of checkpoint to base a price history and time to complete the task;
        based on the time and price threshold select appropriate VM's from the group of replicated VM's;
        set the price, time thresholds and VM's;
}
Checkpoint( ){
        place a checkpoint on the spot instance;
```

```
        calculate next checkpoint based on the failure history of the VM;
        Send the checkpoint to the storage;
}
Recovery( ){
        retrieve the checkpoint information form the storage;
        restart the execution of task;
}
```

## V. Results

There are several available cloud-simulator environments and CloudSim is one of the best of them [18], [19]. Among all classes and packages of the CloudSim, there is no class and package that provides for supporting the implementation of fault-tolerant clouds. Therefore, the creation of an additional package is essential in order to support the implementation of fault-tolerant methods in the cloud computing systems. Consequently, this created package provides the services for fault tolerance by permitting a number of virtual machines of cloud data centers to become faulty. The classes of the package permit the development of algorithms based on fault tolerance which in turn can monitor virtual machines in order to identify failures and then later resolves them. The package can implement both fine grained checkpointing and replication mechanisms. The package offers the capability to measure availability, throughput, time overhead and monetary cost overhead.

As the proposed scheme in this paper can adaptively chooses the replication and fine grained checkpointing mechanisms so it is simple explain that the proposed scheme provides better performance when compared with the earlier techniques which are implemented by considering only checkpointing mechanism or replication mechanism. The performance metrics used in the comparison with a replication based technique and also with a general or fixed interval checkpointing based technique includes availability, throughput, checkpoints overheads and the costs associated with monetary wastages.




## VI. Conclusion

As we know that failures are unavoidable in the environment of cloud computing. To solve this problem, in this paper an adaptive fault reduction scheme that provides reliable cloud computing environment has been proposed. The scheme has two algorithms one for selecting virtual machines to handle users' requests and the other algorithm for selecting the appropriate fault tolerance mechanism. Both replication and fine grained checkpointing methods are encompassed in the scheme. The performance of the scheme is evaluated with a replication based technique and also with a general or fixed interval checkpointing based technique in terms of availability, throughput cloud overheads due to more number of checkpoints and monetary costs. The adaptive nature of the scheme indicates that it will significantly improve the cloud's performance.

## References

[1] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, and I. Brandic, Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, *Future Generation Computing Sytems.*, *vol. 25, no. 6*, pp. 599–616, Jun. 2009.

[2] M. Chen, Y. Ma, J. Song, C. -F. Lai, and B. Hu, Smart Clothing: Connecting human with clouds and big data for sustainable health monitoring, *Mobile Network Appl.*, *vol. 21, no. 5*, pp. 825–845, Oct. 2016.

[3] M. Armbrust et al., Above the clouds: A Berkeley view of cloud computing, Univ. California at Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2009-28. [Online]. Available: http:// www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf.

[4] A. Gómeza, L. Carril, R. Valin, J. Mouriñoa, and C. Cotelo, Fault-tolerant virtual cluster experiments on federated sites using BonFIRE, *Future Generat. Comput. Syst.*, *vol. 34*, pp. 17–25, May 2014.
[5] *The Worst Cloud Outages of 2014*. (Apr. 2016). [Online]. Available: http://www.infoworld.com/article/2606209/cloud-computing/162288-The-worst-cloud-outages-of-2014-so-far.html
[6] K. Bilal *et al.*, Trends and challenges in cloud data centers, *IEEE Cloud Comput. Mag.*, *vol. 1*, *no. 1*, pp. 10–20, 2014.
[7] K. Ganga and S. Karthik, A fault tolerent approach in scientific workflow systems based on cloud computing, in *Proc. Int. Conf. Pattern Recognit., Informat. Mobile Eng. (PRIME)*, Feb. 2013, pp. 378–390.
[8] Z. Zheng, T. C. Zhou, M. R. Lyu, and I. King, Component ranking for fault-tolerant cloud applications, *IEEE Trans. Services Comput.*, *vol. 5*, *no. 4*, pp. 540–550, 4th Quart., 2012.
[9] I. Goiri, F. Julià, J. Guitart, and J. Torres, Checkpoint-based fault- tolerant infrastructure for virtualized service providers, in *Proc. 12th IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Osaka, Japan, Apr. 2010, pp. 455–462.
[10] H. Hui *et al.*, An efficient checkpointing scheme in cloud computing environment, in *Proc. 2nd Int. Conf. Comput. Appl.*, Harbin, China, 2013, pp. 251–254.
[11] S. Limam and G. Belalem, A migration approach for fault tolerance in cloud computing, *Int. J. Grid High Perform. Comput.*, *vol. 6*, *no. 2*, pp. 24–37, Apr./Jun. 2014.
[12] J. Cao, M. Simonin, G. Cooperman, and C. Morin, Checkpointing as a service in heterogeneous cloud environments, in *Proc. 15th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, Shenzhen, China, May 2015, pp. 61–70.
[13] P. Das and P. M. Khilar, VFT: A virtualization and fault tolerance approach for cloud computing, in *Proc. IEEE Conf. Inf. Commun. Tech- nol. (ICT)*, Apr. 2013, pp. 473–478.
[14] S. M. Saranya, T. Srimathi, C. Ramanathan, and T. Venkadesan, Enhanced fault tolerance and cost reduction using task replication using spot instances in cloud, *Int. J. Innov. Res. Sci., Eng. Technol.*, *vol. 4*, *no. 6*, pp. 12–16, May 2015.
[15] E. Zhai, D. Wolinsky, H. Xiao, H. Liu, X. Su, and B. Ford, Auditing the structural reliability of the clouds, Dept. Comput. Sci., Yale Univ., New Haven, CT, USA, Tech. Rep. YALEU/DCS/TR-1479, 2014. [Online]. Available: http://cpsc.yale.edu/sites/default/files/files/tr1479.pdf
[16] Y. Wei, J. Qiu, H. Lam, and L. Wu, Approaches to T-S fuzzy- affine-model-based reliable output feedback control for nonlinear Ito stochastic systems, *IEEE Trans. Fuzzy Syst.*, to be published, doi: 10.1109/TFUZZ.2016.2566810.
[17] Y. Wei, X. Peng, and J. Qiu, Robust and non-fragile static output feedback control for continuous-time semi-Markovian jump systems, *Trans. Inst. Meas. Control*, *vol. 38*, *no. 9*, pp. 1136–1150, 2016.
[18] R. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, *Softw., Pract. Exper.*, *vol. 41*, *no. 1*, pp. 23–50, 2011.
[19] *CloudSim: A Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services*. (Apr. 2016). [Online]. Available: http://www.cloudbus.org/cloudsim

IOSR Journal of Computer Engineering (IOSR-JCE) is UGC approved Journal with Sl. No. 5019, Journal no. 49102.

M. Damodhar. "An Adaptive Fault Reduction Scheme to Provide Reliable Cloud Computing Environment." IOSR Journal of Computer Engineering (IOSR-JCE) 19.4 (2017): 64-73.