IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 1, 2016, accepted October 11, 2016, date of publication November 14, 2016, date of current version January 27, 2017.

Digital Object Identifier 10.1109/ACCESS.2016.2623633

Adaptive Framework for Reliable Cloud Computing Environment

MOHAMMED AMOON
Department of Computer Science, Riyadh Community College, King Saud University, Riyadh 28095-11437, Saudi Arabia
Computer Science and Engineering Department, Faculty of Electronic Engineering, Menoufia University, Menouf 32952, Egypt (mamoon@ksu.edu.sa)

ABSTRACT Cloud computing technology has become an integral trend in the market of information technology. Cloud computing virtualization and its Internet-based lead to various types of failures to occur and thus the need for reliability and availability has become a crucial issue. To ensure cloud reliability and availability, a fault tolerance strategy should be developed and implemented. Most of the early fault tolerant strategies focused on using only one method to tolerate faults. This paper presents an adaptive framework to cope with the problem of fault tolerance in cloud computing environments. The framework employs both replication and checkpointing methods in order to obtain a reliable platform for carrying out customer requests. Also, the algorithm determines the most appropriate fault tolerance method for each selected virtual machine. Simulation experiments are carried out to evaluate the framework's performance. The results of the experiments show that the proposed framework improves the performance of the cloud in terms of throughput, overheads, monetary cost, and availability.

INDEX TERMS Fault tolerance, cloud computing, replication, checkpointing, virtual machines.

I. INTRODUCTION

The current market of Information Technology (IT) has witnessed a considerable change due to the presence of cloud computing, which has become an integral part of most of the businesses [1]. Today, most of the businesses, from single to large enterprises, migrated to cloud computing in order to obtain a high level of productivity by entrusting their IT issues to an expert one. Cloud computing provides comprehensive IT services and solutions for both companies and individual users [2], [3]. They can lease components of the cloud without expending time and money in constructing or buying these components [4]. In cloud systems, computing is introduced as an abstract service over the Internet with hiding the details of implementation [3].

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 certain 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 provides 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 [4], [5].

Private clouds are implemented and maintained by enterprises to provide internal services and they have more flexibility than public clouds but they are more expensive. In hybrid clouds, some portions of computing can be done in a public cloud while other portions can be done internally through the private one [6].

In spite of cloud computing systems used to provide services of computing, they are not perfectly reliable and they could suffer from outages of services due to failures [7]. An outage is defined as the case in which a customer request is not completed in its desired deadline. With the increase of the cloud users, the number of required services increases and

then the probability of outages increases. The main causes of these outages include software failures such as incorrect upgrade, excessive work load, hacking, etc. and hardware failures such as unavailable resource, network failure, power down, etc.

Outages are popular in public clouds in which an enormous number of services is provided to customers with required levels of service quality. In the last decade, many outages have occurred in most famous public cloud environments. In 2013, the home page of Amazon went down for almost an hour, which costs Amazon close to five million US Dollars [8]. In 2014, several Google services such as Gmail, Calendar, Google Docs were stumbled for about an hour. Some servers of Google receive incorrect configurations which cause extensive errors [9]. In 2015, some services of Azure cloud such as virtual machines and websites had more than two and half hours of interruptions across multiple regions [10].

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 [11] 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.

Fault tolerance methods can be reactive or proactive. The main goal of the reactive methods is to reduce the effect of the occurring faults while the goal of the proactive methods is to avoid the occurrence of faults. Reactive methods mainly include replication and checkpointing. Most cloud computing systems depend on reactive methods, especially replication [12].

The replication method assumes that the likelihood of a single VM failure is extremely higher than the occurrence of simultaneous failures of multiple VMs. It allows multiple virtual machines to start simultaneously executing redundant copies of the same request in order to preclude recomputation of it from scratch in a case of failure. Thus, the service can be efficiently provided to customers without affecting their QoS requirements in the presence of failures. In checkpointing, the cloud intermediately saves the execution state of both the currently executed request and the executing VM to a stable storage in order to minimize the recovery time in a case of failure. If a failure occurred, instead of restarting the request's execution from its early start, it will be started from the point in the computation where the last checkpoint was saved [12], [13].

The main contribution of this paper is to present an adaptive framework to cope proactively and reactively with the problem of fault tolerance in cloud computing environments. In order to be proactive, the framework depends on customer requirements and the available information about virtual machines at the scheduling time. Also, the framework employs both checkpointing and replication methods and it dynamically selects the suitable method according to the current conditions of the cloud.

The rest of the paper is arranged 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 framework. In Section 5, results obtained from simulation are presented and the paper concludes in Section 6.

II. BACKGROUND AND RELATED WORK

A. BACKGROUND

The dynamic behavior of the cloud increases the probability of failures. In order to avoid or reduce the effects of these failures, the cloud should apply fault tolerance, which can be reactive or proactive. Reactive fault tolerance methods are applied in order to minimize or omit the influence of failures on monetary and time costs. Replication and checkpointing are the two commonly used reactive methods.

The replication method is based on that the likelihood of failures will be reduced when multiple virtual machines are used to carry out the same customer's request. Recompilation of a request is avoided by performing multiple replicas of the request on different virtual machines at the same time. If a virtual machine fails the cloud can still perform the request within the boundaries of customer's needs. The results of the virtual machine that finishes first are considered and results of other virtual machines are neglected [13].

Checkpointing is the second reactive method. In checkpointing, the status of request's execution is repeatedly saved to a stable and safe storage during execution. In a case of failure, the cloud can continue executing the request starting from the last point at which status was recorded. This will avoid restarting the service of the request from its initial point of execution. Although this can minimize the response time of carrying out a request, more wasted time can result in. This wasted time is due to the recovery of a virtual machine from the failed state if it is the only one that can carry out the task. Nonetheless, the cloud should use this method if there is only a single virtual machine that can carry out the customer's request. The time between two checkpoints is denoted as the checkpoint interval [14].

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 prior to their occurrence. The main goal of these methods is trying to avoid the occurrence of failures and then avoid recovery procedures of the reactive methods. During the scheduling of requests, proactive methods take scheduling decisions according to the prior failure information of the available virtual machines. Therefore, the number of future failures can be reduced and the reliability of the cloud will be improved.

B. RELATED WORK

Fault tolerance is one of the most important issues in distributed computing systems such as grid and cloud computing

systems. In grid computing, there is a lot of fault tolerance work have been done in the literature, whilst a little research has been devoted to the area of cloud computing.

In 2010, Goiri et al. [14] proposed a checkpoint based method that reduces the time needed to store checkpoints. They achieved this through only saving modifications of the read-write regions. They employed the Distributed File System of the Hadoop to save checkpoints. In 2013, Hui et al. [15] proposed a fault tolerant method based on using coordinated checkpoints at the virtual machine level. Their method removes the unavailability due to using coordinated protocols for checkpoint recovery. In 2014, Limam and Belalem [16] defined an adaptive checkpoint method with the aim to remove unnecessary checkpoints or add extra checkpoints according to the current status of the cloud component. Their method increases or decreases the checkpoint interval with fixed rates. In 2015, J. Cao et al. [17] have introduced a uniform fault tolerance method based on checkpointing. Their method supports long jobs and priorities assigned to jobs.

In 2010, Zhao et al. [18] used the replication method in order to propose a fault-tolerant middleware. In 2013, Ganga and Karthik [12] have proposed a replication based method in order to tolerate faults when using scientific workflow systems. Das and Khilar [19] proposed a replication based method to reduce the service time and to increase the system availability. Their method depends on using software variants on several virtual machines to tolerate faults. In addition, it reduces the likelihood of future faults by not scheduling tasks to virtual machines of servers whose success rates are very low. Alhosban et al. [3] introduced a scheme that depends on the prediction and planning. A method of recovery is elected to be applied when faults occurred. The selection depends on failure history, user requirements and service weight and criticality. Methods that can be selected are replication and retry.

In 2012, Zheng et al. [13] have proposed an algorithm that can select a fault tolerance method for each virtual machine. All methods that the algorithm can select from are variations of the replication method such as parallel and multiversion. In 2015, Saranya et al. [20] presented and evaluated a method based on both replication and resubmission of tasks. Their method depends on a priority assigned to each task depending on tasks length, their deadline and the out-degree of each task. In 2015, Liu and Wei [21]] proposed a replication based algorithm that considers the failures of both hardware and software.

The analysis of literature shows that most of the previous work done are fundamentally based on using a single fault tolerance method, either replication or checkpointing. There is a little work done that considers using both of the two methods together to tolerate faults in cloud computing systems. Also, most of the existing replication based work considers a static or fixed number of replicas and they do replication for all virtual machines in the cloud, which is not an economic approach. In the case of checkpointing, most of the proposed work assumes fixed or fixed change of the length of the checkpoint interval during the execution of the customer requests or jobs. There is a little work done that considers the adaptive length of the checkpoint interval. So, there is a need for a framework that considers both replication and checkpointing methods and selects the number of replica or checkpoints in an adaptive manner.

III. PROBLEM DESCRIPTION

Cloud services are provided either as storage services or computing services. Dropbox, iCloud and Google are examples of storage services and Amazon EC^2 and Microsoft Azure are examples of computing services. In order to be served, a customer submits his service request to the cloud provider along with the requirements needed for his request. The provider negotiates with the customer in order to determine both the quality of service and the price. If the customer accepts, the provider will prepare the cloud virtual machine that can carry out the request and the service will start.

Most of the cloud resources are not primarily designed to accomplish the economic objective of the cloud. These resources are collected into various virtual machines to accomplish customer requests. So, it is expected that numerous failures will occur that will protract the time expected to carry out the customer requests and this will exhaust the cloud resources. For customers, they will not get their services in the time expected. For the cloud, failures will lead to loss of cloud resources and then money. This will lead to a considerable impact on the reliability, credibility and reputation of the cloud [22]. Thus, it is heavily required to implement fault tolerance methods in cloud computing systems in order to alleviate or omit the influence of failures on the performance of the cloud.

Replication of both data and applications is the method used by most of the current cloud computing systems. It is applied in Amazon S3 by storing data objects on multiple storage units. The iCloud can rent infrastructure services from Amazon's EC2 or Microsoft's Azure to fulfill the replication. In spite of that, cloud outage reports refer to the point that the reliability is still insufficient [23]. Applying fault tolerance methods in clouds faces the following challenges:

1. The cloud can have only a single copy of the virtual machine that can carry out the request of the customer. Also, the cloud may have multiple virtual machines that can carry out the customer's request, but only one is available and the others are busy in performing other services or they are out of service. So, replication method cannot be applied.
2. The number of replicas should not be static or fixed because this will lead to a poor influence on the cloud. This is due to the fact that additional virtual machines will be used to carry out the same service. However, these virtual machines can be used to carry out other customer services. Thus, the cloud will lose profit charges.

3. It is not economical to implement replication for each service or virtual machine. Replication should only be applied for services that are allocated to the most valuable virtual machines that will have a great impact on the performance of the cloud if they fail. Determining the most valuable virtual machines is a great challenge.
4. In the checkpointing method, determining the length of the checkpointing interval is a major challenge. Checkpointing with fixed checkpoint interval could lead to redundant checkpoints that consume cloud resources and increases checkpointing latency.

In order to cope with the first challenge, checkpointing method is involved in our framework beside replication. Our framework allows the cloud to choose either checkpointing or replication in order to achieve fault tolerance. In order to address the second challenge, a replication algorithm that adaptively determines the number of replicas of an application is proposed. For the third challenge, the percentage of profit gained by the cloud when using the virtual machine is involved in determining the number of replicas required for each virtual machine. For the fourth challenge, an algorithm that adaptively determines the length of the checkpointing interval is proposed. The algorithm assumes that the length of the checkpointing interval must not be fixed during the execution of the customer's task. The algorithm considers the failure probability of virtual machines to calculate the next checkpointing interval.

IV. CLOUD ARCHITECTURE

Cloud computing environments should have the ability to receive, perform, monitor and control customers' requests. The cloud should be reliable in order to provide its services within the limits of customer requirements. This section describes the proposed framework which enables the cloud to be reliable. As shown in Figure 1, the architecture of the framework assumes the cloud consists of three main layers: application, virtual and physical layers. One function of the application layer is to allow customers to interact with the cloud. Also, it performs the scheduling of customers' requests to the virtual machines in the cloud. In addition, tolerating faults is the responsibility of the application layer. In order to perform these functions, the structure of the application layer comprises four modules:

1. *Service Inspector*: This module is responsible for ensuring the achievement of customer's QoS requirements. In this paper, the considered QoS requirements include time and monetary costs. A customer can submit his request to the cloud through this module along with the QoS requirements. The module asks the Status Database module for the availability of appropriate VMs that can carry out the customer request and gets a reply. If the reply indicates the presence of appropriate VMs that can carry out the request within customer's requirements, the Service Inspector will accept the request and it will deliver it to the Scheduler module. Otherwise, the request will be discarded.

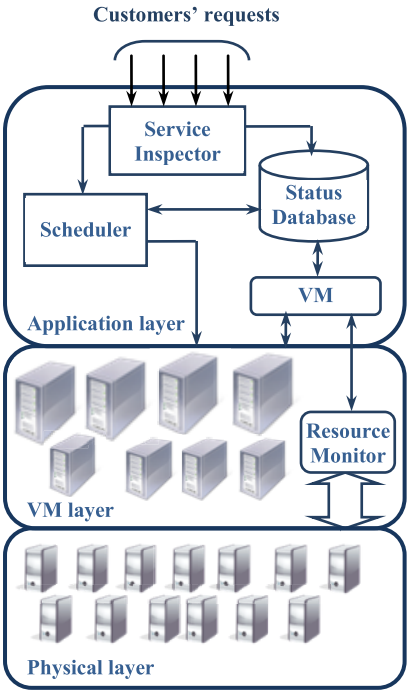

FIGURE 1. The level architecture of a cloud computing system.

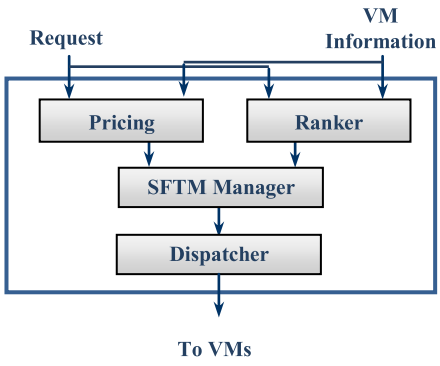

FIGURE 2. Scheduler components and their interactions.

2. *Scheduler*: The main function of the Scheduler is to assign each request to the suitable virtual machine that can perform it within the limits of customer requirements. Also, Scheduler has the responsibility of determining the charge of serving the request. In addition, Scheduler has the responsibility of fault tolerance. In order to do its responsibilities, the Scheduler module should contain the following components: Ranker, Pricing, Scheduling and Fault Tolerance Manager (SFTM) and Dispatcher. Figure 2 illustrates the interactions between the main components of the Scheduler. The main role of the Ranker is to determine the most valuable VMs in the cloud (see subsection 4.2). It receives customer's request with QoS requirements from the Service Inspector and contacts the Status Database module in order to get information about the virtual machines that can accomplish the request. Based on this information, it prepares a list

Algorithm 1 SFT Algorithm

```
Input: c_ju is the cost required by the customer u for request
       j,
       τ_ju is the deadline time required by the customer u
       for request j,
       c_ji is the estimated cost if the request j is executed
       by VM_i,
       τ_ji is the estimated time if the request j is executed
       by VM_i,
j = 1;
While (there are requests not served)
        {
        For each request j do
              {
              Find a list of VMs that can carry out j;
              For each VM_i in the list do
                  If (c_ji > c_ju||τ_ji > τ_ju)/*VM_i cannot
                  serve j*/
                              remove VM_i from the
                              list;
                  If (list is not empty) /*The request
                  can be served */
                        {
                        Sort the VMs list ascending
                        based on c_ji × τ_ji;
                        If (there is more than one VM
                        in the list)
                              Replication is selected;
                        Else
                              Checkpointing is
                              selected;
                  Else {
                        Send "Request cannot be
                        served" to the QoS Controller;
                        End the algorithm for j;
                        }
                  j++; /* next request */
              }/* For end*/
        }/* While end*/
```

of VMs that can fulfill the time and monetary requirements of the customer's request. Pricing component determines the charge the customer should pay for the service. SFTM implements Algorithm 1 in order to select the appropriate fault tolerance method for the virtual machine assigned to each request. The algorithm selects either checkpointing or replication based on information about virtual machines. Dispatcher delivers the requests of customers to the selected VMs.

3. *Status Database*: It represents the central repository of information about all virtual machines in the cloud such as computing capacity, storage capacity, price, usage history and failure history.
4. *VM Monitor*: The main function of this module is to observe the performance of the virtual machines in the cloud. It notifies the Status Database to update the record of a VM in a case of the failure or the recovery of that VM. In addition, this module has the responsibility for forming or reforming virtual machines of the cloud. It has virtualization software used to form unique and isolated virtual machines using the cloud physical resources.

VM layer is the second layer of the cloud. It contains virtual machines of the cloud and each virtual machine is formed using one or more physical resources. Also, each physical resource may be shared and used by multiple virtual machines. Furthermore, different VMs can be emulated on a single physical resource in order to satisfy the requirements needed by customer requests. This layer has a module called Resource Monitor. The main function of this module is to observe the performance of the physical resources of the cloud and to notify the VM monitor with changes occurred. Changes include resources leaving the cloud or new resources joining the cloud. Upon these changes, Resource Monitor can reform affected virtual machines.

The third layer of the cloud is the physical layer and it contains hardware and software resources of the cloud. Resources are the real operators in the cloud.

A. SFT ALGORITHM

Algorithm 1 is called the Selecting Fault Tolerance (SFT) algorithm and it is proposed with the objective to select the appropriate method for tolerating faults in the cloud computing system. The algorithm is implemented in the SFTM component of the Scheduler module. In order to achieve its objective, the algorithm depends on using customer's requirements and the available information about virtual machines. First, the algorithm prepares a list of virtual machines that can carry out the customer's request and satisfies the customer's requirements. The customer's requirements considered by the algorithm include both time costs and monetary costs. Thereafter, the algorithm selects checkpointing method if there is only a single VM in the list. Otherwise, the algorithm selects replication method.

B. REPLICATION ALGORITHM

Replication is applied when there are multiple and available virtual machines in the cloud that can carry out the customer's request. However, it is a central challenge to define the optimal number of replicas. In addition, it is not an economical approach to perform replication for all virtual machines [24]. So, we only need to replicate requests executed on the most valuable virtual machines that will have a great impact on the performance of the cloud if they fail.

In order to determine the most valuable VMs in the cloud, VMs should be ranked according to their value and influence 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 can determine its probability to fail. For each virtual machine, failure history can be

represented by the number of failures occurring, failure time, the time between failures and failure types. The need of a virtual machine to a fault tolerance method is determined by failure probability. As the value of the failure probability becomes high, the need for applying fault tolerance methods rises.

In general, the occurrence of random failures is a stochastic process [25] and Jump Linear Systems (JLSs) can be used to model it because they involve event driven and time evolving techniques. The process is based on the time period between two consecutive faults. In clouds, this time period is a random variable following general probability distributions and the process is often called semi-Markov process. The jump linear system of the semi-Markov process is known as semi-Markovian JLS with time-varying transition rates [26], [27].

In this work, the failure probability of a virtual machine is assumed to follow Poisson distribution. This means that the number of failures in any two different or disjoint periods of time is independent over the time change. The failure probability distribution of a virtual machine i in a given time interval can be expressed as:

$$F_i(X) = \frac{e^{-\mu}\mu^x}{x!} 0 \leq F_i(X) \leq 1 \text{ and } x = 0, 1, 2, \ldots, n, \quad (1)$$

where $X(x_0, x_1, x_2, \ldots, x_n)$ represents the number of failures happened in a given time period and μ is the average number of failures in the given time period for a virtual machine i. The value of μ is given by:

$$\mu = \frac{f_i}{T_i/\tau_{ji}}, \quad (2)$$

where f_i is the number of failures of a virtual machine i and T_i is the period of time in which f_i failures have occurred. τ_{ji} represents the estimated time when request or application j is executed on virtual machine i. Thus, the probability of one failure ($x = 1$) to take place during the execution of a request is given by:

$$F_i(x_1) = \mu e^{-\mu}. \quad (3)$$

The virtual machine profit, denoted as P_i, represents the percentage of cloud profit gained through the usage of virtual machine i in performing requests. The value of the virtual machine i to the cloud is determined by its profit. The greater the profit of a virtual machine the more its value.

The rank of a virtual machine is computed by the Ranker component of the scheduler. The Ranker obtains the values of failure probability and profit of virtual machines from the Status Database. Thereafter, it calculates the rank of each virtual machine using the formula:

$$R_i = \mu e^{-\mu} \times P_i, \quad (4)$$

where R_i is the rank of virtual machine i, $\mu e^{-\mu}$ is the probability of a failure to take place and P_i is the profit of virtual machine i.

The fixed number of replicas is not an efficient choice in cloud computing environments because additional virtual machines will be used to carry out the same request. However, these virtual machines can be used to carry out requests of other customers. Thus, profit charges will be wasted. Also, it is not economically to implement replication for each request or for each 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
- $F_i(X)(k)$, $k = 0, 1, 2, \ldots, n$, are integers such that $0 <= F_i(X)(k) <= 1.0$ and $F_i(X)(0) < F_i(X)(1) < \ldots < F_i(X)(n)$
- $P_i(y)$, $y = 0, 1, 2, \ldots, m$, are the percentage of cloud profit gained by virtual machine i such that $0 <= P_i(y) <= 100$ and

$$P_i(0) < P_i(1) < \ldots < P_i(m)$$

- $Rep(l)(w)$, $l = 0, 1, 2, \ldots, n$ and $w = 0, 1, 2, \ldots, m$, are integers

For $(a = 0; a < n; a{+}{+})$
{
For $(b = 0; b < m; b{+}{+})$
{
If $(F_i(X)(a) \leq F_i(X) < F_i(X)(a+1)$ and $P_i(b) \leq P_i < P_i(b+1))$

$$Rep = Rep(a)(b);$$

}
}

Algorithm 2 is the replication algorithm proposed in this paper in order to adaptively determine the number of replicas of a request. The number of replicas will not be fixed for all requests or virtual machines. In order to adaptively determine the number of replicas, the operation of the algorithm depends on both the failure probability and the percentage of cloud profit gained by the virtual machine assigned to carry out the customer's request. As either the failure probability or profit percentage of a virtual machine increases the need for more replicas increases. Consequently, virtual machines with higher values of profit or failure probability have higher fault-tolerance needs and then higher priority of replication than other VMs.

C. CHECKPOINT ALGORITHM

Distributed systems, such as grid computing systems, have widely used checkpointing as a reactive fault tolerance method to alleviate the impact of failures when occurred. Moreover, most cloud computing systems implement replication techniques. However, from the perspective of the cloud service provider, replication results in profit loss due to allocating extra components to execute the replicas of a request, particularly these components may be useful for other requests. Also, from the perspective of customers,

replication leads to time loss due to waiting for components that execute replicas to be free from executing other requests. So, the main advantage of using checkpointing over replication is to preserve the computing resources of the cloud to other customers' requests and to reduce the profit loss because of using replication.

Checkpointing interval and latency are the two parameters that strongly influence a checkpointing algorithm. The checkpointing interval represents the time between a checkpoint and the next checkpoint. Checkpointing latency is the time consumed in saving a checkpoint. In the case of small checkpoint interval, there will be a large number of checkpoints. This large number of checkpoints will heavily consume cloud resources when saving checkpoints and thus high checkpointing latency results in. Moreover, long checkpoint interval leads to a small number of checkpoints and then a considerable part of the request should be recomputed in the case of failure. This small number of checkpoints will slightly consume cloud resources when saving checkpoints and thus low checkpointing latency results in.

So, determining the length of the checkpointing interval is the major challenge for a checkpointing technique. Fixed interval leads to redundant checkpoints that consume cloud resources and increase checkpointing latency. So, the main objective of our work is to develop an algorithm that adaptively determines the length of the checkpointing interval. Algorithm 3 assumes that the length of the checkpointing interval must not be fixed during the execution of the customer's task. The algorithm calculates the next checkpointing interval at the time of the current checkpoint. It is calculated based on the failure history of the VM on which the task is executed. In the case of a poor failure history, the algorithm will shorten the checkpoint interval. Moreover, the algorithm will prolong the checkpoint interval in case of good failure history.

Algorithm 3 Checkpoint Algorithm

- τ_{ji} : The execution time of task j on VM i
- τr_{ji} : The remaining execution time of task j on VM i
- $F_i\,(x_1)$: Failure probability of VM i
- $F_i\,(x_0)$: Probability of no failure of VM i
- h: Checkpoint interval
- z: Number of failures during the task execution

Calculate $F_i\,(x_1) = \mu e^{-\mu}$;
For each task j allocated to VM i do
{
$z = 0$; $h = \tau_{ji} \times F_i\,(x_z)$; //Initial checkpoint interval
$\tau r_{ji} = \tau_{ji}$;
Start execution of j on i;
do
{

$$\tau r_{ji} = \tau r_{ji} - h;$$

If failure occurred then
{
z++;
$\tau r_{ji} = \tau r_{ji} + h$;
$h = h(1 - F_i\,(x_z))$; // decrease checkpoint interval
Restore last checkpoint;
Restart execution from $\tau_{ji} - \tau r_{ji}$;
}
At time $\tau_{ji} - \tau r_{ji}$ perform a checkpoint;
$h = h(1 + F_i\,(x_z))$; // increase checkpoint interval
Resume execution;
}While ($\tau r_{ji} \neq 0$)
}

V. RESULTS

There are many available cloud-simulator environments and CloudSim is one of the most of them [28], [29]. Among all classes and packages of the CloudSim, there is no one that supports the implementation of fault-tolerant clouds. So, the creation of an extra package is needed in order to support the implementation of fault-tolerant methods in the cloud computing systems. This created package provides services of fault tolerance through allowing some virtual machines of cloud data centers to be faulty. The classes of the package allow the development of fault tolerant based algorithms that can monitor virtual machines in order to detect failures and resolve them. The package can implement both checkpointing and replication techniques. The package provides the ability to measure throughput, availability, time overhead and monetary waste overhead.

The cloud used in our experiments is generated with 100 heterogeneous virtual machines that are connected with fast Ethernet technology (100Mb/s). The number of data centers used in each experiment ranges from 5 to 10. Each data center contains 4 hosts. The size of each host's memory is 10 GB and the storage is 2TB. The processing capacity of computational units in each host is assumed to be in the range from 1000 to 10000 MIPS. The number of customer requests ranges from 500 up to 2500 requests. Each virtual machine has a memory of 4 GB and one computational unit. The size of data required for each request processing is randomly selected from 10 MB up to 1 GB. The price cost of the cloud computing unit is assumed to be in the range from \$0.1 to \$10

We evaluate the performance of our proposed framework by comparing it with the checkpointing based algorithm proposed in [17], named optimal checkpoint interval (OCI) algorithm, which is based on using variable checkpoint intervals. Different simulation experiments have been conducted with a variable number of customers' requests. The performance metrics used in the comparison include throughput, availability, checkpoints overheads and the amount of monetary waste.

Figure 3 shows the results of the throughput comparison between the proposed framework and OCI algorithm. The number of requests is shown in the x-axis and the results of throughput, measured in requests per hour, are plotted as columns. In general, the throughput of both the proposed

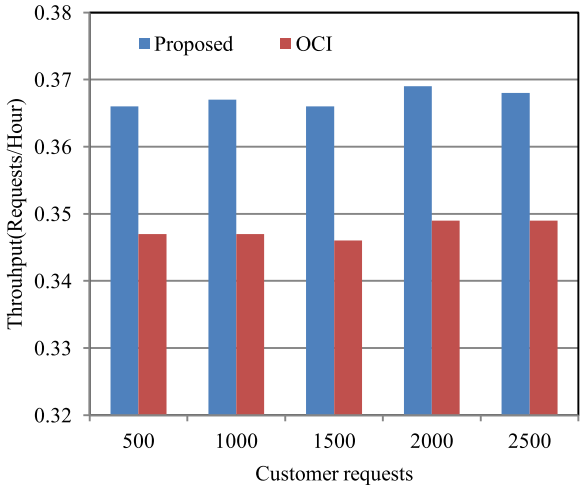

FIGURE 3. Throughput Comparsion.

framework and the OCI algorithm increases with the increase in the submitted customers' requests. The figure clearly shows that the proposed framework has a better throughput than the OCI algorithm. This is because the proposed framework has less turnaround time than the OCI algorithm. This is attributed to the fact that the proposed algorithm considers the failure probability as a criterion when selecting virtual machines to carry out requests. On the other hand, the OCI algorithm considers the number of failures. This will make our proposed framework less prone to fail and more reliable than the OCI algorithm.

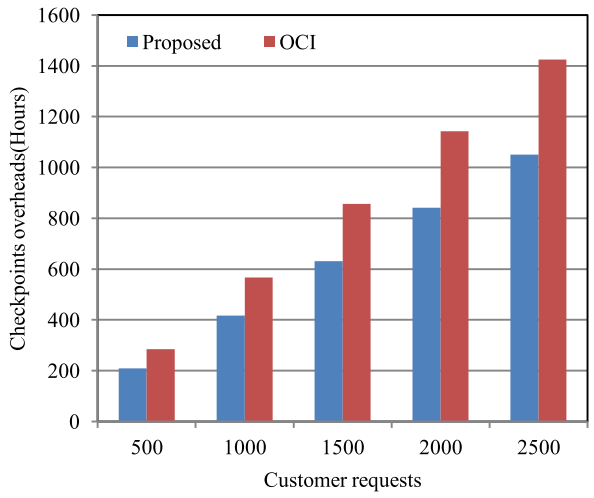

FIGURE 4. Overheads comparsion.

Figure 4 illustrates the comparison of overheads between the proposed framework and OCI algorithm. The figure shows that the overheads of the proposed framework are less than that of the OCI algorithm. This is because our proposed framework adaptively determines the length of the next checkpointing interval while the OCI algorithm changes it with constant rates. Thus, the proposed framework eliminates the unnecessary checkpoints and then overheads are reduced.

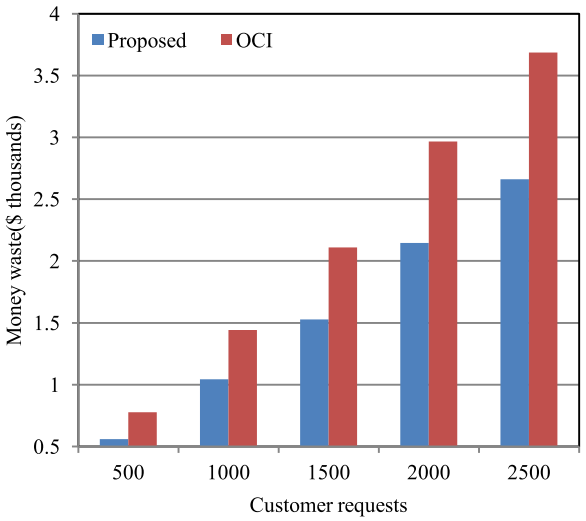

FIGURE 5. Monetary cost comparsion.

Figure 5 illustrates the comparison of monetary waste between the proposed framework and OCI algorithm. We can see that the proposed framework has a lower monetary waste than the OCI algorithm. This is due to that the proposed framework has a less number of failures and a less number of checkpoints than the OCI algorithm. This will save the resources of the cloud for other customer requests and thus money is saved.

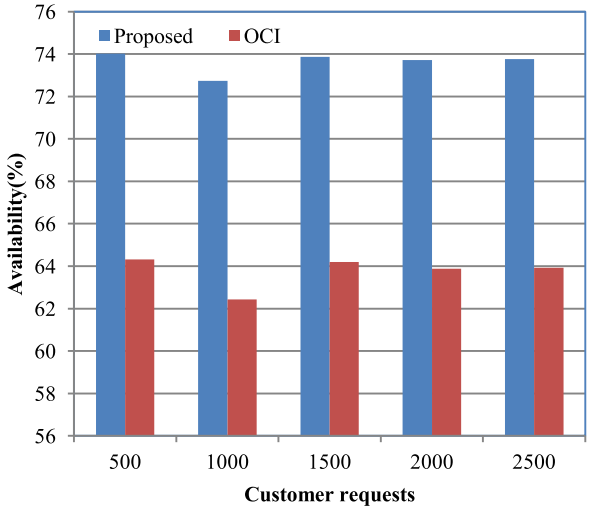

FIGURE 6. Availability comparsion.

The availability of a cloud is the percentage of the service time and failure time. The service time is the operational time of the cloud per a certain period of time. Figure 6 shows the comparison of availability between the proposed framework and OCI algorithm. The figure shows that the proposed framework provides better availability than the OCI algorithm. This is due to the adaptive checkpoint interval that

our proposed framework provides which helps to decrease the number of failures occurring.

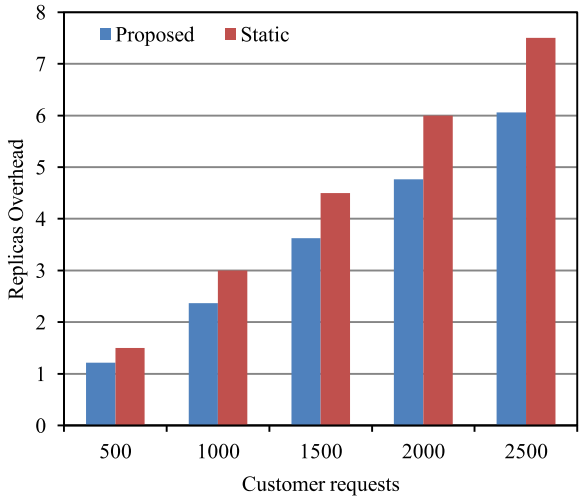

FIGURE 7. Overhead comparsion.

Also, we evaluate the performance of our proposed framework by comparing it with the replication based algorithm proposed in [21]. As this algorithm considers a static or fixed number of replicas, we will denote it the static algorithm. Figure 7 illustrates overheads' comparison between the proposed framework and static algorithm. The term of overheads represents the number of replicas the cloud should perform. The figure shows that the overheads of the proposed framework are less than that of the static algorithm. This is because our proposed algorithm considers an adaptive number of replicas that can dynamically change for each request according to the current conditions of the virtual machine assigned to perform the request. On the other hand, the static algorithm considers a fixed number of replicas regardless the current conditions of the virtual machines assigned.

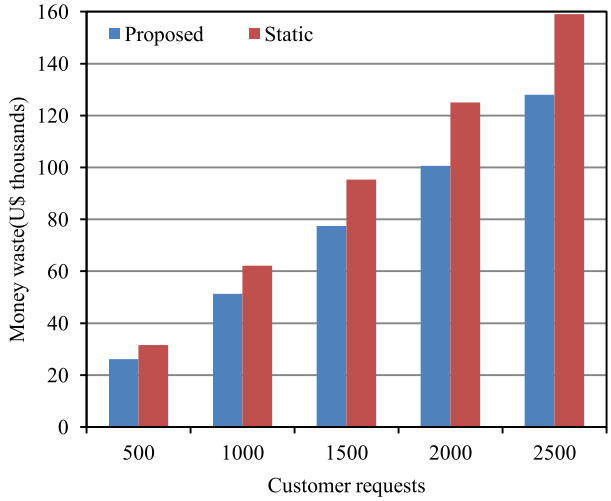

FIGURE 8. Money waste comparsion.

Figure 8 shows the monetary waste comparison between the proposed framework and static algorithm. We can see that the proposed framework has a lower monetary waste than the static algorithm. This is because the proposed framework only replicates the most valuable virtual machines and not all virtual machines as the static algorithm. This will decrease the number of virtual machines consumed in the replication. Thus, profit charges will not be lost.

TABLE 1. Performance improvement.

Metrics	Percentage of improvement
Throughput	5%
Overheads	23%
Monetary cost	24%
Availability	13%

From the above results of the experiments, it is shown that the proposed framework improves the performance of the cloud in terms of throughput, overheads, monetary cost and availability. The adaptive nature of the proposed framework gives it superiority over the other related ones. This adaptive nature appears when determining the number of replicas for virtual machines or when calculating the checkpoint intervals. Improving throughput will improve the number of services the cloud can serve in the same time and then the profit of the cloud increases. Improving the overheads leads to saving resources of the cloud for other customers. This will reduce the waiting time for customer requests. Improving the amount of monetary waste will allow the cloud provider to enhance the services of the cloud through continuous maintenance and new resources added. Improving availability of the cloud will reinforce the trust of the customers.

VI. CONCLUSION

Failures are unavoidable in cloud computing environments. To treat this issue, an adaptive framework for tolerating faults in cloud computing environments has been proposed in this paper. The framework has one algorithm for selecting virtual machines to carry out customers' requests and another algorithm for selecting the suitable fault tolerance method. Both replication and checkpointing methods are included in the framework. The performance of the framework is evaluated with a replication-based algorithm and also with a checkpointing-based algorithm in terms of throughput, cloud overheads, monetary cost and availability. Experimental results indicate that the proposed framework improves the cloud's performance as shown in Table 1.

In the future work, we will include investigations about applying our framework and the well-established fault detection and reliable control methods for complex industrial processes, such as developed in [30] and [31], in the cloud computing environment. Also, we will provide more consideration to the migration of data centers and tasks between them.

ACKNOWLEDGMENT

This work was supported by King Saud University, Deanship of Scientific Research, Community College Research Unit.

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 Generat. Comput. Syst.*, 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 Netw. Appl.*, vol. 21, no. 5, pp. 825–845, Oct. 2016.

[3] A. Alhosban, K. Hashmi, Z. Malik, and B. Medjahed, "Self-healing framework for cloud-based services," in *Proc. Int. Conf. Comput. Syst. Appl.*, May 2013, pp. 1–7.

[4] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: The montage example," in *Proc. ACM/IEEE Conf. Supercomput.*, Austin, TX, USA, 2008, Art. no. 50.

[5] 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.

[6] L. Wang, M. Kunze, J. Tao, and G. Laszewski, "Towards building a cloud for scientific applications," *Adv. Eng. Softw.*, vol. 42, no. 9, pp. 714–722, Sep. 2011.

[7] 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.

[8] *The Worst Cloud Outages of 2013*. (Apr. 2016). [Online]. Available: http://www.infoworld.com/slideshow/107783/the-worst-cloud-outages-of-2013-so-far-221831

[9] *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

[10] *Assessing Cloud Infrastructure Provider Performance in 2015*. (Apr. 2016). [Online]. Available: http://searchcloudcomputing.techtarget.com/feature/Assessing-cloud-infrastructure-provider-performance-in-2015

[11] K. Bilal *et al.*, "Trends and challenges in cloud data centers," *IEEE Cloud Comput. Mag.*, vol. 1, no. 1, pp. 10–20, 2014.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] W. Zhao, P. M. Melliar-Smith, and L. E. Moser, "Fault tolerance middleware for cloud computing," in *Proc. 3rd Int. Conf. Cloud Comput. (CLOUD)*, Miami, FL, USA, Jul. 2010, pp. 67–74.

[19] P. Das and P. M. Khilar, "VFT: A virtualization and fault tolerance approach for cloud computing," in *Proc. IEEE Conf. Inf. Commun. Technol. (ICT)*, Apr. 2013, pp. 473–478.

[20] 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.

[21] Y. Liu and W. Wei, "A replication-based mechanism for fault tolerance in mapreduce framework," *Math. Problems Eng.*, vol. 2015, 2015, Art. no. 408921

[22] R. Jhawar, V. Piuri, and M. Santambrogio, "Fault tolerance management in cloud computing: A system-level perspective," *IEEE Syst. J.*, vol. 7, no. 2, pp. 288–297, Jun. 2013.

[23] 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

[24] E. Bauer and R. Adams. *Reliability and Availability of Cloud Computing*. Hoboken, NJ, USA: Wiley, 2012.

[25] 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.

[26] 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.

[27] Y. Wei, J. Qiu, and H. R. Karimi, "Quantized filtering for continuous-time Markovian jump systems with deficient mode information," *Asian J. Control*, vol. 17, no. 5, pp. 1914–1923, 2015.

[28] 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.

[29] *CloudSim: A Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services*. (Apr. 2016). [Online]. Available: http://www.cloudbus.org/cloudsim

[30] L. Li, S. X. Ding, J. Qiu, Y. Yang, and Y. Zhang, "Weighted fuzzy observer-based fault detection approach for discrete-time nonlinear systems via piecewise-fuzzy Lyapunov functions," *IEEE Trans. Fuzzy Syst.*, to be published, doi: 10.1109/TFUZZ.2016.2514371.

[31] J. Qiu, S. X. Ding, H. Gao, and S. Yin, "Fuzzy-model-based reliable static output feedback control of nonlinear hyperbolic PDE systems," *IEEE Trans. Fuzzy Syst.*, vol. 24, no. 2, pp. 388–400, Apr. 2016.

MOHAMMED AMOON received the B.S. degree in electronic engineering in 1996 and the M.Sc. and Ph.D. degrees in computer science and engineering from Menofia University in 2001 and 2006, respectively. He has been appointed as a Lecturer with Menoufia University in 2006 and an Associate Professor in 2014. He is currently an Associate Professor with Menoufia University and King Saud University. His research interests include distributed computing, grid computing, cloud computing, green IT, scheduling and resources allocation, and agent-based systems.