ISSN : 2231-0711

Torki Altameem | International Journal of Computer Science Engineering and Technology(IJCSET) | Dec 2014 | Vol 4, Issue 12,395-399

A Replication-Based and Fault Tolerant Allocation Algorithm for Cloud Computing

Torki Altameem

Dept. of Computer Science, RCC,
King Saud University,
P.O. Box: 28095 – 11437
Riyadh-Saudi Arabia.

Abstract— **The very large infrastructure and the increasing demand of services of cloud computing systems lead to the need of an effective fault tolerant allocation technique. In this paper, we address the problem of allocating user applications to the virtual machines of cloud computing systems so that failures can be avoided in the presence of faults. We employ job replication as an effective mechanism to achieve efficient and fault-tolerant cloud. Most of the existing replication-based algorithms use a fixed number of replications for each application which consumes more cloud resources. We propose an algorithm to determine adaptively the number of replicas according to the fault rate of cloud virtual machines. The proposed algorithms have been evaluated through simulation and have shown better performance in terms of turnaround time and throughput.**

Keywords— Fault tolerant, Replication, Cloud Computing, Fault rate.

I. INTRODUCTION

Cloud computing is an internet based computing solution which is considered as the next step in the technology evolution of distributed computing. It provides a comprehensive solution for delivering IT as a service and it facilitates scalable and cooperative sharing of resources among different organizations. The cloud enables on-demand access to applications from anywhere in the world, without considering their implementation details [1]. Resources include storage, processors, platforms, or application services. The flexibility of cloud computing is a function of allocating resources on demand [2], [3].

In clouds, resources of different physical machines can be grouped into Virtual Machines (VMs). These VMs can be started and stopped on-demand on to meet service requests. This provides maximum flexibility to configure various partitions of resources for different specific requirements of user requests [4].

VMs can fail to do their work due to their heterogeneity and usage for longer periods of time. Failure of VMs has a great impact on scalability, performance, profit and consumer trust. Fault tolerance is an approach where a cloud computing system continues to work successfully even if there is a fault[5].

Although cloud computing has been widely adopted by the industry, fault tolerance is still a main research challenge to be fully addressed [6].Because of the very large infrastructure of cloud and the increasing demand of services an effective fault tolerant allocation technique for cloud computing is required[2].

The main mechanisms used in implementing fault tolerance in cloud computing include checkpointing and replication. Checkpointing is the ability to save the state of a running application to a stable storage. In case of any fault, this saved state can be used to resume execution of the application from the point in computation where the check-point was last registered instead of restarting the application from its very beginning [7]. In this paper, job and application will be used interchangeably.

Replication is based on the assumption that the probability of a single VM failure is much higher than of a simultaneous failure of multiple VMs. It avoids job recomputation by starting several copies of the same job on different VMs. With redundant copies of a job, the cloud can continue to provide a service in spite of failure of some VMs carrying out job copies without affecting the performance[8].

Cloud applications must have dynamic fault-tolerant services that detect faults and resolve it. These services enable applications to carry on their computations in case of failure without terminating applications. Also, these services must satisfy the minimum levels of quality of service (QoS) requirements of applications such as the deadline to complete the applications, the number of computing VMs, the type of the platform and so on.

In this paper, the main contribution is to develop a replication based algorithm for allocating applications of users to the VMs of the cloud computing system. The algorithm selects VMs according to the finishing time of applications rather than the response time. The algorithm generates the number of replicas dynamically. This means the number of replicas will not be fixed for all the applications.

This paper is organized as follows: section 2 briefly explains related work of tolerating faults in cloud computing systems. Section 3 elaborates the proposed algorithm. Section 4 augments results and discusses the performance of the proposed algorithm. Section 5 presents our conclusion.

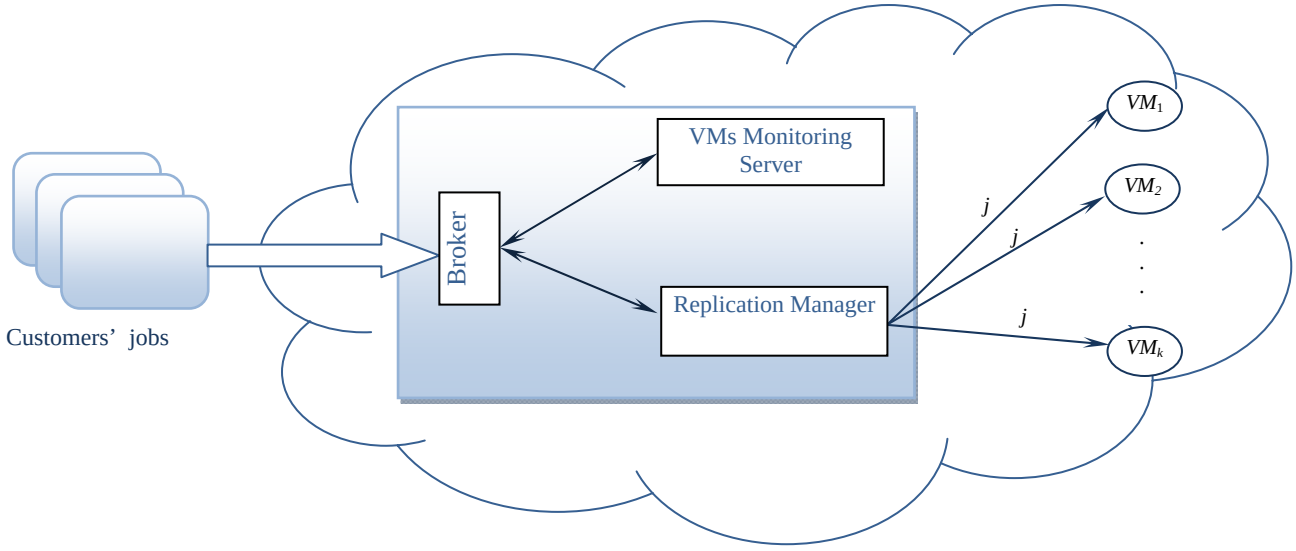

Figure 1. The components of the cloud computing system.

II. RELATED WORK

In [3], K. Ganga and S. Karthik discussed the fault tolerance techniques and classified them as proactive and reactive techniques. Proactive techniques predict the failure and replace the suspected resources from the other working resources. Reactive techniques try to reduce the effect of failures when occurs such as checkpointing, replication and resubmission. They focused on applying job replication on scientific workflow systems.

Alhosban et al. [1] provides a technique to dynamically evaluate the performance of services based on their previous history and user requirements. Their technique uses a prediction and planning approach and it consists of two phases. In the first phase, the fault likelihood of the service is assessed. In the second phase, they built a recovery plan to execute in case of fault(s) and they calculated the overall system reliability based on the fault occurrence likelihoods assessed for all the services.

Das and Khilar [2] have proposed a reactive model that integrates fault tolerance with cloud virtualization. Their model depends on using success rate of the computing nodes and virtualization and it includes the support of load balancing. They have used replication mechanism to achieve the fault tolerance.

Jhawar, Piuri and Santambrogio [9] have presented an approach toward transparently delivering fault tolerance on the applications deployed in virtual machine instances. They have presented an approach for realizing generic fault tolerance mechanisms as independent modules, validating fault tolerance properties of each mechanism, and matching user's requirements with available fault tolerance modules to obtain a comprehensive solution with desired properties. Also, they have designed a framework that allows the service provider to integrate its system with the existing cloud infrastructure and provides the basis to generically deliver fault tolerance as a service.

Reviewing literatures reveals that all previous works are mainly based on using the response time as the main criteria when selecting VMs that can execute a cloud application. There is no work done in the area of cloud computing that considers the finishing time of applications on the cloud VMs. In this paper, an algorithm that depends on the finishing time of jobs or applications on cloud VMs is presented and evaluated.

Also, our algorithm uses the mechanism of replication to tolerate faults of VMs if occurred. Replication provides an efficient way to guarantee the completion of jobs according to the QoS required by the user. In cloud computing, most of the existing replication based algorithms uses a fixed number of replicas. This fixed number of replicas can lead to the use of extra VMs in executing user applications. These extra VMs may be needed by other waiting applications. Thus, cloud will lose the monetary benefit of these VMs. So, there is a need to a way to provide a dynamic number of replicas to preserve the cloud resources.

III. THE PROPOSED ALGORITHM

The main purpose of the proposed algorithm is to improve the performance of the cloud through minimizing both the time spent by the application in the cloud and the effect of failure if occurred. The algorithm depends on selecting VMs that have the earliest finishing time for user applications. Also, it depends on using the replication mechanism to generate multiple copies of the same application to be executed on multiple VMs, simultaneously.

The components of the cloud computing system used in our paper are shown in Figure 1. These components include the broker, the VM monitoring server, the replication manager and the cloud VMs. Consumers or users submit their applications or jobs along with their QoS requirements to the cloud through the cloud portal. The jobs will be inserted in the broker queue. The broker will receive a job from the broker queue along with its required QoS. Then, it will ask the VM Monitoring Server for a list of suitable VMs for executing the job. The server will reply with a list of VMs that can perform the application along with their expected finish times for the user's application. The broker will sort this list according to the finish time of the application on each VM. The first VM in the sorted list,

which is the VM with the earliest finish time, is selected as the main VM to execute the job.

The finish time of a VM *i* for a job *j* is defined as:

$$FT(j,i) = t(j,i) + ST(j,i), \qquad (1)$$

where *t*(*j*, *i*) is the execution time of job *j* on VM *i* and *ST* (*j*, *i*) is the time at which job *j* will start execution on VM *i*. The value of the *ST*(*j*, *i*) is summation of the execution time of all the jobs assigned to the VM *i* and executed or to be executed before job *j* and it can be defined by:

$$ST(j,i) = \sum_{k=1}^{j-1} t(k,i). \qquad (2)$$

Since the selected main VM may fail, the system will choose some other VMs from the list on which copies of the job will be executed. Replicating a job can help dealing with failure; then when one VM fails, the adverse effect on performance of the application it runs can be reduced if replicas complete without failure.

Now, the broker will ask the replication manager to determine the number of replicas. The number of replicas should not be high in order to avoid cloud overloading. The number of replicas is based on the failure rate of the main VM and the QoS requirements. The failure rate of a VM is a representation of the failure history of it. Assume f_i is the number of times a VM *j* failed to complete jobs and n_i is the total number of jobs assigned to the VM *j*. The failure rate of a VM *i* is defined by:

$$fr_i = \frac{f_i}{n_i}, where\ 1 \le fr_i \le 0. \qquad (3)$$

When providing his QoS requirements, the customer determines whether he needs replication of his application or not. This can be achieved through a factor called *rep* whose value is given by the customer to the cloud server provider. If the value of *rep* = 0 then the customer does not need to replicate his application. If the customer needs to replicate his application he will set *rep* = 1.

If rep =1, Replication Manager uses the failure rate of the main VM to determine the number of replicas for the customer's applications. It checks the value of *fr* of the main VM. If the value of *fr* is less than *k* it will add an extra replica for the application. If the value of *fr* is greater than or equal *k* it will add two extra replica for the application. The value of *k* is determined by the cloud service provider according to the abundance of VMs that can execute the application without affecting allocation of other applications. Figure 2 shows the main steps of the proposed algorithm.

```
For each application(job) j submitted to the cloud
Begin
    Receive a job with QoS requirements from the portal;
    Request a list of suitable VMs the job from the VM monitoring
server;
    Receive a list of suitable VMs from VM monitoring server;
    Compute FT(j, i) for each VM i;
    Sort the list in an ascending order according to the FT(j, i) ;
    Determine the main VM as the first one in the list;
    Compute the fr for each the main VM;
    R_j = 0;/* R_j number of replications of application j*/
    If(rep==1)
        If(0 ≤ fr ≤ k)
            R_j = 1;
        Else
            R_j = 2;
     EndIf
    Retrieve next job as j;
End
```

Figure 2. The proposed system's operation.

IV. Results and Analysis

In this section, the performance of the proposed algorithm is compared against the performance of a replication-based algorithm that depends on using the response time in selecting VMs and uses a fixed number of replicas. The comparison is performed within cloud data centers with varying load and reliability. In the simulation experiments, the number of applications submitted ranges from 100 to 500. The number of VMs in the grid is assumed to be 1000.

1. A. Throughput

Throughput is one of the most important standard metrics used to measure the performance of fault tolerant systems [10]. It is used to measure the ability of the cloud to accommodate applications. Throughput is defined as:

$$Throughput(n) = \frac{n}{T_n}, \qquad (4)$$

where *n* is the total number of applications submitted and T_n is the total amount of time necessary to complete the *n* applications. Throughput acts as an indicator of the cloud's profit. When the throughput increases the profit of the cloud will increase.

Figure 3 and Figure 4 show the throughput comparison of the proposed finish time based algorithm with a response time based algorithm [8] for different number of applications submitted.

In this experiment, the numbers of submitted applications by the customers to the cloud are 100, 200, 300, 400 and 500. The percentage of faults injected in the cloud is 10% in Figure 3 and 20% in Figure 4.

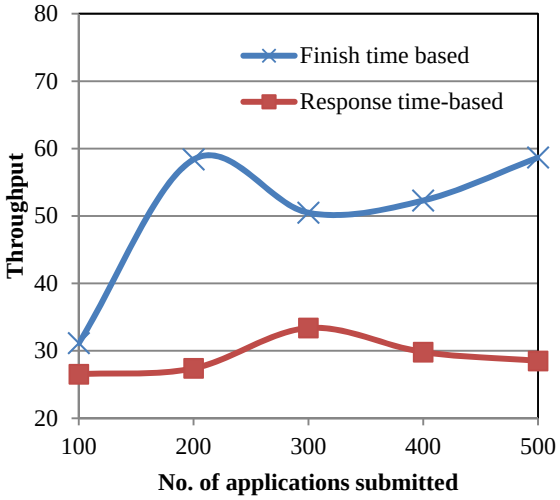

Figure 3. Throughput Comparison with 10% injected faults.

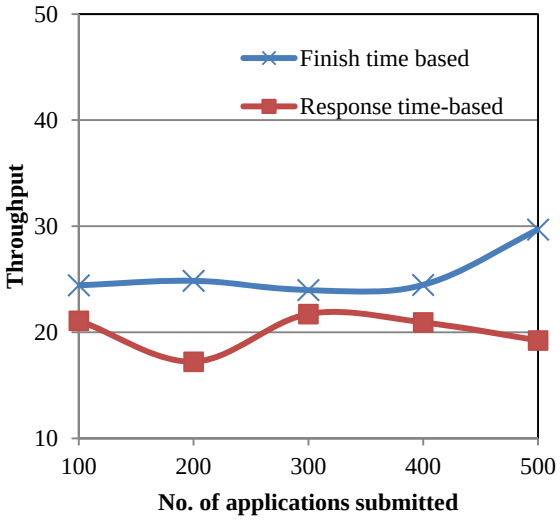

Figure 4. Throughput Comparison with 20% injected faults.

The figures shows that the throughput of the proposed finish time based algorithm is better than the throughput of the response time based algorithm for the whole range of job numbers. This is because that one VM can have a good response time when executing an application but it may have a delay to start that application. This delay is due to that the VM may have a lot of work to do before starting the execution of the application. On the other hand, another VM with a worst response time but has a little work to do can start that application and finish it earlier. Also, it is shown from the figures that the value of throughput in Figure 4 is less than its value in Figure 3 at the same number of applications submitted. This is due to that in Figure 4 the number of VMs faults is greater than the number of VMs faults in Figure 3.

2. B. Turnaround Time

Turnaround is an important parameter for evaluating the performance of distributed computing systems. It is the most important parameter users pay attention to. It can be defines as the interval between application submission and application completion.

Figure 5 depicts the comparison of the proposed finish time based algorithm with a response time based algorithm [8] for different percentages of faults injected in the cloud. The percentages of faults injected are from 5% to 25%.

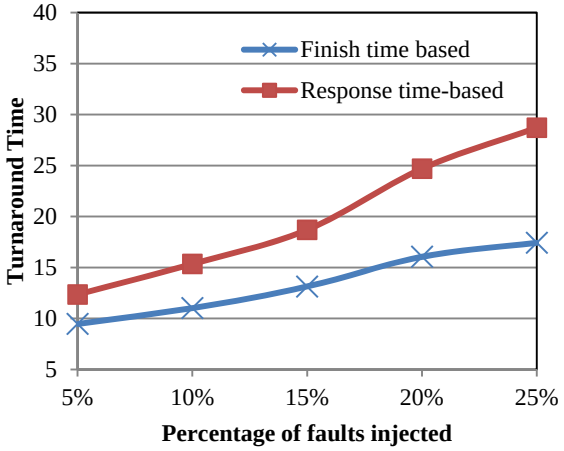

Figure 5. Turnaround time comparison with 500 applications submitted.

In general, the turnaround time resulting from using the two algorithms increases with the increase in the number of applications submitted. The figures show that the proposed algorithm has better turnaround time than the other algorithm for different application sizes.

This is because in the response time based algorithm the number of faulty VMs is more than that of the proposed algorithm. This will lead to more delay time resulting from those main VMs that have the best response time fail to execute the assigned applications. As the number of failed VMs increases, the delay time will increase and thus the turnaround time for executing user applications will increase. On the other hand, the proposed system selects the VMs which are less prone to fail. This will lead to a small number of faulty VMs and lower delay times than the other algorithm.

V. CONCLUSION

In this paper, we presented and evaluated a fault tolerant allocation algorithm for cloud computing systems that uses the replication mechanism. The algorithm depends on using the finishing time in selecting VMs. Also, the algorithm computes the number of replica for an application according to the fault rate of the VM allocated to execute the application. The performance of the proposed algorithm is evaluated under different conditions using metrics such as throughput and turnaround time. From results of experiments, it can be concluded that the proposed algorithm provides better performance.

REFERENCES

[1] A. Alhosban et al, "Self-healing Framework for Cloud-based Services," Proceedings of the 2013 Int'l Conf. on Computer Systems and Applications, May 27-30

[2] P. Das, P. Mohan Khilar, "VFT: A Virtualization and Fault Tolerance Approach for Cloud Computing," Proceedings of the IEEE Conference on Information and Communication Technologies (ICT 2013), Kanyakumari, Tamil Nadu, in press, 2013, pp. 473-478.

[3] K. Ganga and S.Karthik, "A Fault Tolerent Approach in Scientific Workflow Systems based on Cloud Computing," Proceedings of the IEEE International Conference on Pattern Recognition, Informatics and Mobile Engineering, February 21-22, 2013, pp.117-122.

[4] R. Buyyaa et al," Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Generation Computer Systems, vol. 25, 2009, pp. 599-616.

[5] D. Singh, J. Singh and A. Chhabra, "High Availability of Clouds: Failover Strategies for Cloud Computing using Integrated Checkpointing Algorithms", IEEE International Conference on Communication Systems and Network Technologies, 2012.

[6] P. Gupta and S. Banga, "Review of Cloud Computing in Fault Tolerant Environment with Efficient Energy Consumption," International Journal of scientific research and management, Vol. 1, Issue 4, 2013, pp. 251-254.

[7] G. Belalem and S. Limam,"Fault Tolerant Architecture to Cloud Computing Using Adaptive Checkpoint," International Journal of Cloud Applications and Computing, Vol. 1, Issue 4, October-December 2011, pp. 60-69.

[8] Z. Zheng, T. C. Zhou, M. R. Lyu and I. King, "Component Ranking for Fault-Tolerant Cloud Applications," IEEE Transactions On Services Computing, Vol. 5, No. 4, October-December 2012, pp. 540-550.

[9] R. Jhawar, V. Piuri and M. Santambrogio, "Fault Tolerance Management in Cloud Computing: A System-Level Perspective," IEEE Systems Journal, Vol. 7, No. 2, June 2013, pp. 288-297.

[10] M. Huda, H. Schmidt and I. Peake, "An agent oriented proactive fault-tolerant framework for grid computing," Proc. of International Conference on e-Science and Grid Computing, Melbourne, Australia, pp. 304-311, Dec. 5-8, 2005.