JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Reliability Aware Mechanism to ensure increased fault tolerance using Throttle Load Balancer

Rudhrakshi Bhagat
M.Tech Scholar
Sri Sai Group Of Institutes

Er. Satnam Singh Dub
Assistant Professor
Sri Sai Group Of Institutes

Dr. Kamal Kishore
Associate Professor
Modern Group Of Colleges, Pandori Bhagat

Abstract. Cloud computing provides resources to the machines having limited resources associated with them. Cloud resources although are infinite but still lacks as the users of the cloud increases day by day. Early completion of job assigned to the virtual machine considerably solves the problem of starvation and availability. This paper proposed a mechanism to handle tasks from various sources submitted to the cloud. The mechanism consist of two phases: first phase is used to monitor the virtual machines that lead to selection of optimal VM having sufficient resources and second phase allocate the resources to the tasks. Continues monitoring process using throttle load balancing, also checks the load on individual machine. In case load increases, migration to the next VM is initiated. The parameters considered in the proposed system follow reliability metric. The metric enhancement is the main objective of proposed system. The overall mechanism suggested through the proposed system fall under proactive fault tolerance. The simulation is conducted within MATLAB and cloudsim 4.0 and evaluation parameters are load balancing degree, throughput and execution cost. Performance enhancement by 6% is observed which is significant proving worth of study.

Keywords: Cloud computing, Monitoring, Throttle load balancing, energy efficiency, throughput, execution time

1. Introduction

Cloud computing provides the mechanism to share resources[1] on pay per use basis. Cloud service should ensures resource availability as and when required. Lack of resources within the cloud computing[2] environment can cause starvation. This problem becomes vigorous as complex jobs are submitted to the brokers within the cloud. Strategies are devised over to rectify the issue of lack of resources within the cloud computing. The main strategy employed nowadays includes advance reservation.

Advance reservation[3] ensures that jobs should enter into the system as and when resources are available and demands of the jobs can be fulfilled. Advance reservation thus can prolong the affect of faults and failures that occur when load is increased over the server and virtual machine. To perform advance reservation[4], nodes are maintained within the cloud architecture by the broker. This broker calculates the demands[5], [6] associated with each cloudlet. Cloudlet requirements are matched against the VM configuration. Cloudlets that match the desired requirements are added into the list, otherwise they are discarded. By using this method of advance reservation, throughput [7] is reduced significantly. The proposed literature works towards improving the performance of this system by using the strategy of handling the jobs without discarding them and providing the resources to the jobs as and when they are available ensuring higher throughput.

In a cloud based resource management system, cloud provider act both as service provider as well as consumer. The prime concern of service provider is to enhance revenues and utilization of resources by increasing consumers. The consumers here could be federation members[8] or regular cloud members. The role of resource management is prominent in the scenario and leads to availability of resources for the federation members and regular cloud users. The objective of federation member and cloud user is satisfied through effective resource management[4], [9] ensuring the successful collaboration of both as federation member and cloud users are essential entities and objectives of both should be satisfied appropriately.

Resource provisioning within cloud must consider reliability metrics as useability may or may not always enhances performance. Resources provided through VMs may be of no use in case VM fails. In order to address this issue [10], [11]suggested VM migration, replication and fault tolerant protocol in high performance cloud. Cloud with single provider is handled with the fault tolerance strategies but least amount of work is done towards the reliability aspect within federated cloud. To this end, disaster management must be incorporated within the federation of cloud to enhance the execution if there arises an occurrence of dynamic provisioning of resources from various cloud service organizations. [12]

This study concentrates on resource provisioning along with advance reservation mechanism within cloud. The importance of service reliability metric for service provisioning is considered for evaluation[13]. Overall goal is to enhance throughput by using proactive fault tolerance approach using VM monitoring to configure the problem of VM allocation to the jobs. Resources allocation should be such that it can reduce the overall resource consumption associated with the jobs. Rest of the paper is organised as under: section 2 gives the literature survey, section 3 gives the proposed system, section 4 gives the experimental setup, section 5 gives the performance analysis and result and section 6 conclusion and future scope, section 7 gives the references.

2. Literature Survey

As of late, numerous techniques are used to determine the optimal approach for minimising the resource utilization along with load balancing associated with the cloud data centers. Overall cost associated with the system must also be reduced.[14] A hierarchical copy approach was proposed by Z.Liu in which allocation of resources is minimized. With the reduction in the resource consumption, the load balancing is improved by significant manner. Y.Lin proposed energy efficient fault tolerance replication process to conserve resources and energy. Fault tolerance with the proposed literature is high. Response delay is considered additional parameter which is reduced also. Joint optimization mechanism was proposed by X.Jin to optimize resource utilization and energy consumption. Resource utilization was improved through said mechanism by incorporating VM consolidation mechanism[15]. J.Sekhar proposed live vm migration to minimize the execution time of the jobs. The live vm migration states that machines both at the source and destination end does not switched off even during migration there by reducing downtime and migration time.

In classic data center model [16] the architecture of cloud computing is divided into data centers, where data centers are further divided into hosts and hosts are partitioned into virtual machines having powerful computing power. These virtual machines are used to allocate the resources to the jobs. The association and importance of the data centre can take various variations and expecting to lessen inactivity and energy utilization. One of the fundamental difficulties in distributed computing[17] is to expand the accessibility of computational resources, while limiting framework control utilization and operational costs. This [18], [19] presents a power productive resource distribution calculation for errands in distributed computing server farms. The created approach depends on hereditary calculations which guarantee execution and versatility to a huge number of undertakings. Resource designation is performed considering computational and organizing necessities of undertakings and streamlines energy consumption.[20]

From the literature it is concluded that sufficient amount of work has been done and research continuously evolving towards minimization of resource consumption to decrease the cost associated with the overall execution of cloudlet or jobs.

3. Proposed System

Fault tolerance mechanism which is proactive in nature is used to ensure high throughput and load balancing degree. This process is used in order to make our system resilient against the faults occurring within the system. To prove the worth, faulty environment is considered and performance is compared against the existing system. This algorithm ensures that cloudlet according to capacity of VM is allotted at any given time. If more request groups are present than the number of available VM's at data centre, allocate incoming request in queue basis until the next VM becomes available.

Objectives

1. The reliability is considered as the measure of number of cloudlets which are successfully executed by virtual machines to the number of cloudlets submitted, which must be high and is achieved using throughput.
2. Minimizing the cost associated with the cloudlet execution using strategy of resource maximization.
3. Using load balancing to make overall system fault resilient.

Initialization Module

- First of all, the cloudlets are recived and virtual machines are generated according to capacity of the host. In this process, the allocation of various cloudlets to VMs in data centre has been done on the basis of optimized resources. Optimised resources will be in the form of maximum availability of RAM, bandwidth and processing element.
- Example- if we have 128MB of RAM and process uses 64MB of it, then total left resources will be 64MB.
- The optimized resources are found using following equation:
  OR= TRU + CRF
  Where a TRU total resource utilized and OR is optimal resources. The CRF is calculated as:

  $CRF=[r\ (1+r)^{n}/\ (1+r)^{n}-1]$

Where CRF (Cumulative resource factor) is based on the overall resources used in system.
r= probability of resource
n= resource count

Resource Maximization and selection

- The VM will be selected in order of resource maximization
- The requirement of cloudlet will decide the selection of virtual machine. If virtual machine is available then it will be allocated to the cloudlet otherwise it will be added to the queue.

- **Changes to the existing system with Monitoring (Monitoring Module)**
- This will execute TLB() methods. If VM is not available then it will be added in to the queue. It will wait until all execution by VM is done. In this work insertion into queue is done by increasing rear of the queue.

- Algorithm of TLB

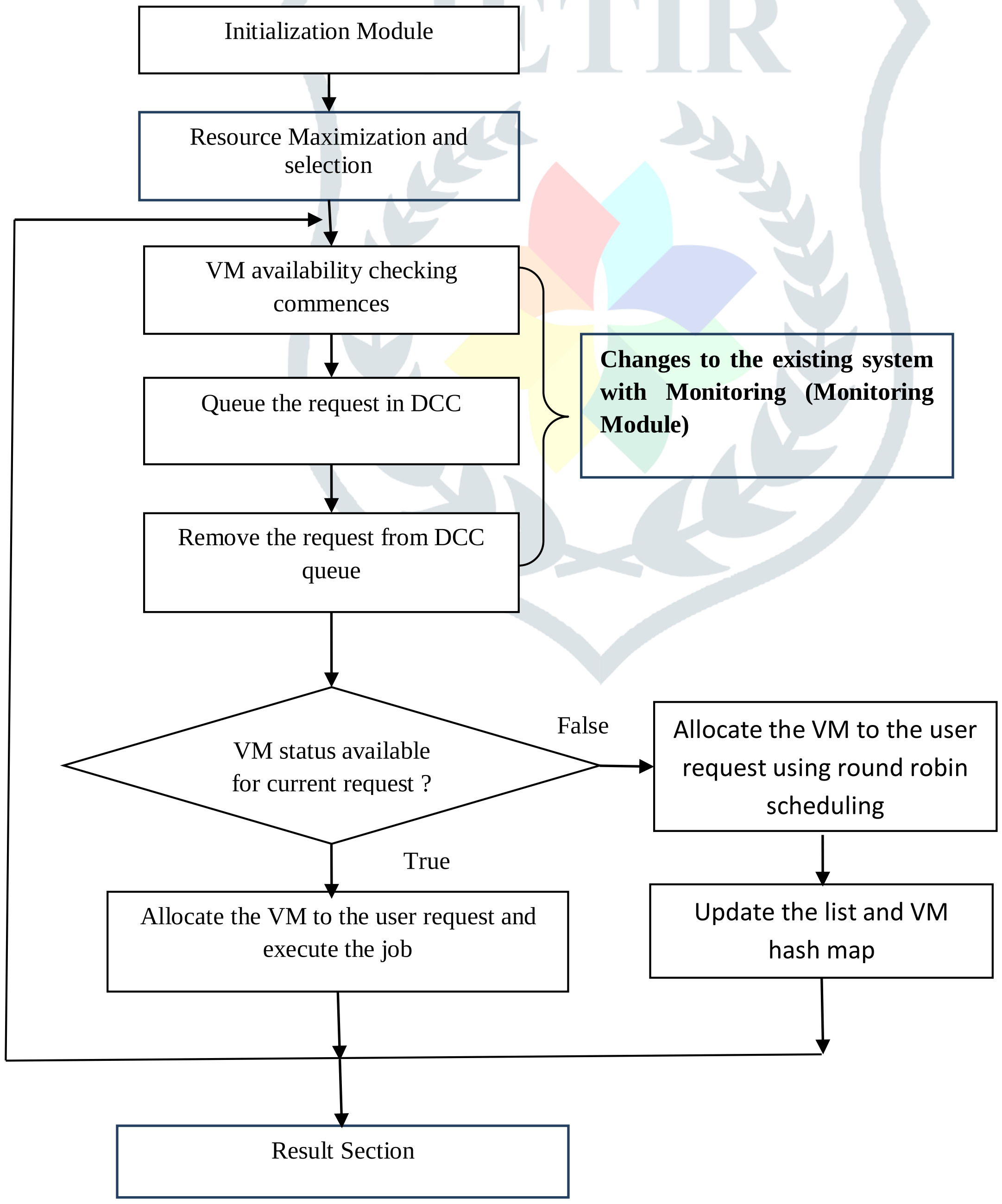
{Initialize all the VM allocation status to AVAILABLE in the VM state list; Initialize hash map with no entries;
While(new request are received by the Data Centre Controller)

- Do {
  Data Center Controller queue the requests; Data Centre Controller removes a request from the beginning of the queue;
- If(hash map contain any entry of a VM corresponding to the current requesting user base && VM allocation status == AVAILABLE) {
  The VM is reallocated to the user base request;}
  Else {
  Allocate a VM to the user base request using Round Robin Algorithm; Update the entry of the user base and the VM in the hash map and the VM state list;}
  } }
- After that TLBD process executes. In this after all the execution by VM is done than the jobs from queue is fetched and VM execute that job. The de queue operation is performed that would remove the job from queue by decreasing front and allocate job to the VM list.

Result Section

- Results in terms of parameters such as Load Balancing factor, throughput
  $TH = I / T$
  where TH is throughput , I is total jobs, T is time required to complete work.
  LB= TRA/ Total request + TVA/ Total cloudlet
  Where TRA indicates total resources available, where TVA indicated total VM Available

This methodology is incorporated at early stage of load allocation to tackle the issue of faults at later stage occurring due to load enhancement on the virtual machine. The flowchart of proposed methodology is as given below:


Initialization Module
Resource Maximization and selection
VM availability checking commences
Queue the request in DCC
Remove the request from DCC queue
Changes to the existing system with Monitoring (Monitoring Module)
VM status available for current request ?
False
True
Allocate the VM to the user request using round robin scheduling
Update the list and VM hash map
Allocate the VM to the user request and execute the job
Result Section

Resource maximization indicates resource gathering phase in which tasks which are phantom or zombies are eliminated from the memory and resources are made free. In cloudsim this can be achieved by the use of delete keyword. All the cloudlets maintained within the cloudlet list are removed and fresh allocation with new environment can commence. Once the procedure of checking vm availability is complete, resource is allotted to the cloudlet. In case vm is not available then cloudlet is added within the queue. This queue is maintained at the broker end and the cloudlet is fetched from the queue as and when availability of VM takes place. Result and performance analysis shows considerable improvement in terms of throughput, cost and load balancing.

4. Experimental Setup

The experimental setup consists of creation of data centers, hosts, virtual machines, brokers and cloudlets. The experiment is simulated cloudsim 4.0. Cloud environment contains the host which is partitioned into virtual machines. The number of virtual machines that are required in the proposed system is 25 for each host. Quantity of datacenters defined in the environment is 2 and every datacenter is further divided into 2 hosts. Each host is partitioned into 25 virtual machines which derived there resources from the host. Jobs for allocation known as cloudlets are 50. These cloudlets are alloated to the virtual machine considering the resources with each virtual machine. Time shared policy is implemented to share the virtual machines among the jobs or task. 512 MB RAM is used with each Virtual machine. Image size is of 20,000BM, Number of processing elements with each VM is 1. Total of 100 cloudlets with 40000 image size are created to demonstrate the simulation process.

This setup is used to evaluate the validity of the proposed system.

5. Performance Analysis and Results

The results are obtained in terms of cost, load balancing degree and total throughput achieved through the system.

5.1 Cost

In adaptive framework technique, three algorithms which were concurrently used: selecting fault tolerance, checkpointing and replication. These algorithms were used on requirement basis. In case of light faulty environment, replication mechanism is considered, in relatively heavy faults checkpointing technique comes into picture, cost is determined by considering number of virtual machines utilized to execute the cloudlets. In case cloudlets are distributed over more virtual machines than high cost is encountered.

$$Total_{Cost} = \sum_{i=1}^{n} Cost\_VM_i$$

Equation 1: Total cost associated with cloudlet

The cost will be encountered as resource being utilized. The amount of time the resource is used along with the number of resources used directly impacts the cost. Cost is evaluated by looking at the cloudlets allocation. Higher the allocation more will be the cost. This cost evaluation is considered with and without the faulty environment. First of all we give the result without fault injection.

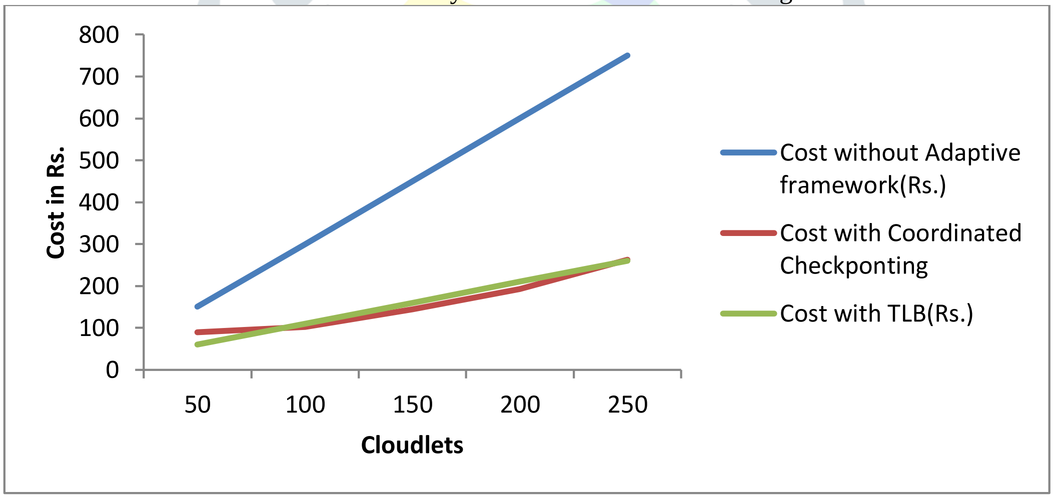


Figure 1: Cost determination when no fault is injected.

In second situation, proposed system and existing system is passed through the faulty environment. Faults are injected using the random distribution mechanism. The proposed system deviates however with the small amount from current solution. Cost associated with each VM is uniformly distributed. This is given as :

$$Cost_i=3.0$$

This cost is associated with each VM. The evaluation of cost when cloudlet is assigned to vm is given as

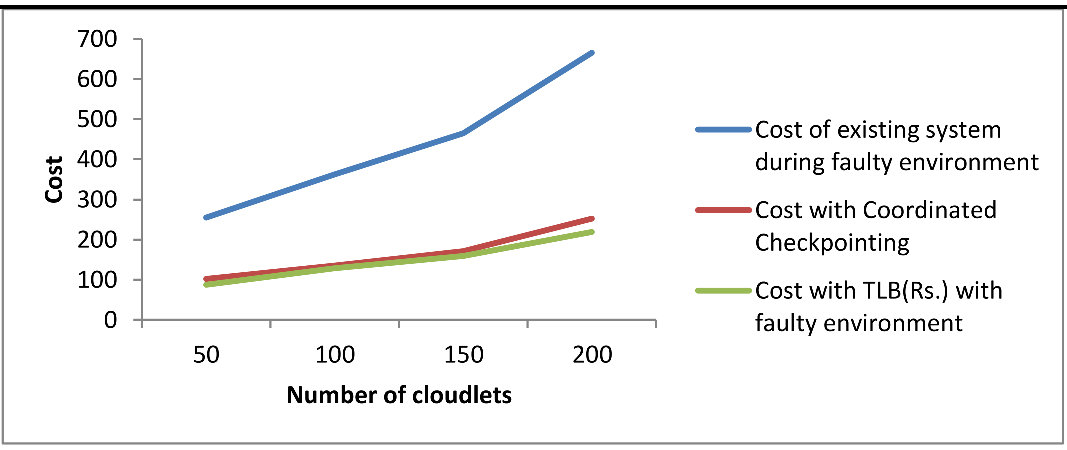


Figure 2: Cost in case of faulty environment

5.2 Load Balancing Degree

Load balancing degree indicates the resource utilization. Higher the load balancing degree, resource utilization is optimised. It is observed in the range of 0 to 1. Evaluation of load balancing degree is critical since it will enhance or degrade the performance of the system. Load balancing degree is evaluated as under.

$$Load_{Balance_i} = \sum_{i=1}^{n} \frac{VM_{Load_i}}{N}$$

Equation 2: Load balancing degree evaluation

VMload indicates the total cloudlets executed by the current vm. N is the total number of cloudlets.
In case fault is not injected, load balancing degree of proposed and existing system is given as under:

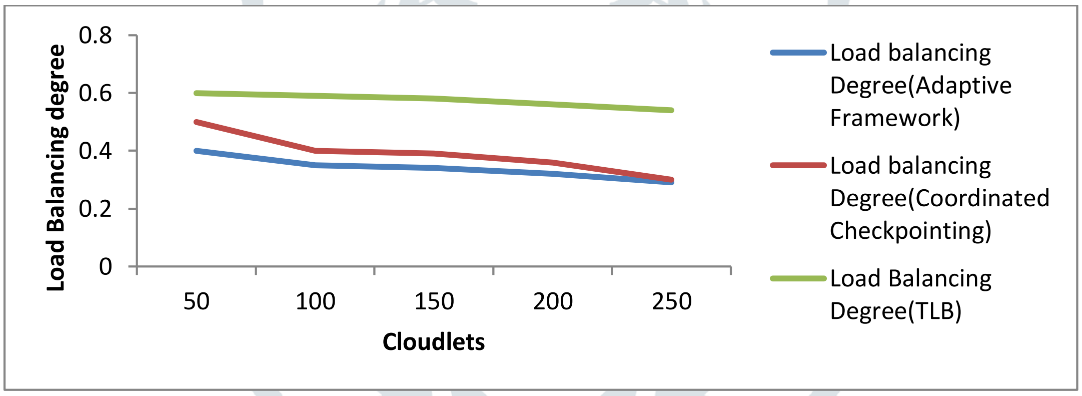


Figure 3: Load balancing degree associated with fault free environment

When faults are introduced the performance of existing and proposed system degrades. The deviation in result is minor in case of proposed system showing worth of study. The load balancing degree affected by fault injection is given as under:

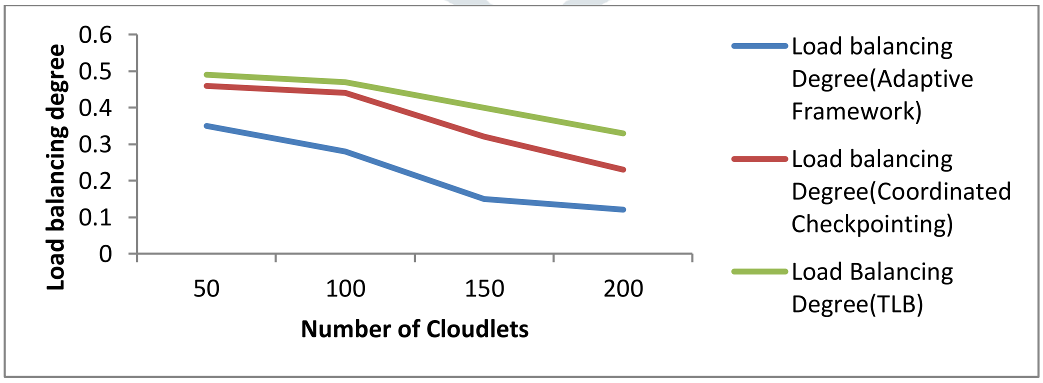


Figure 4: Load balancing degree when faults are introduced

5.3 Throughput

Another parameter evaluated to indicate better result corresponding to proposed approach is throughput which is total execution of cloudlets. The total execution of cloudlets out of submitted cloudlets are evaluated using the following equation

$$Throughput = \sum_{i=1}^{n} \frac{Executed_{Cloudlets_i}}{Total_{Cloudlets_i}}$$

Equation 3: Throughput evaluated equation

Result obtained through the proposed system is considerably better as compared to existing literature. In proposed literature, a special queue is maintained at the cloudlet execution phase. In case VM is not available, broker put the cloudlet into the queue. As and when resource become available, broker fetch the resource from the queue and allots the job to the available virtual machine. This reduces resource consumption and enhances load balancing degree. Plots corresponding to throughput are given as under

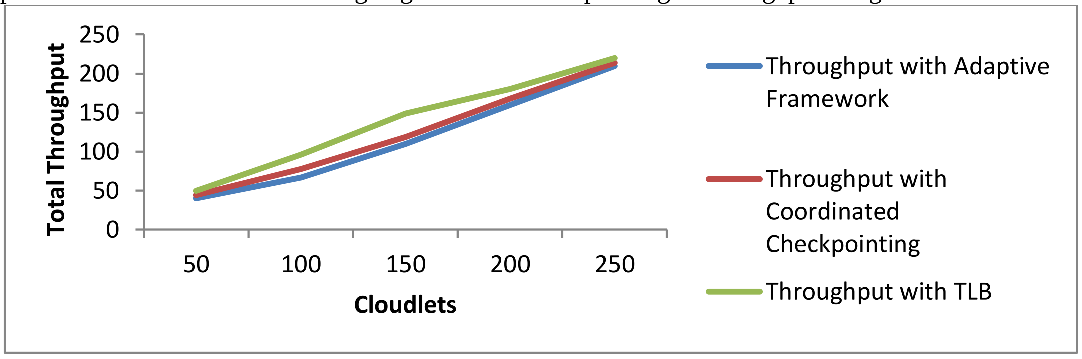


Figure 5: throughput without considering faulty environment

As faults are injected, performance in terms of total output is reduced. In other words, number of cloudlets execution deteriorate as the faults becomes prominent within the existing and proposed system.

Results in terms of plot in fault prone environment is given as under

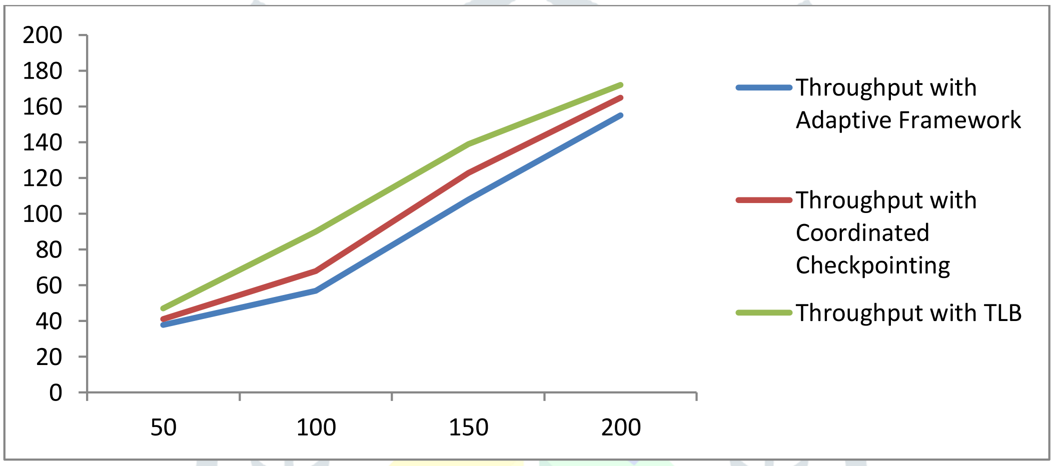


Figure 6: Throughput in case of faulty environment

Result shows improvement by the significant manner by the considered approach.

6. Conclusion and Future Scope

Resource utilization becomes a hot concern among researchers. This paper is an extension of mechanism used to ensure reduced resource utilization to conserve cost and enhances efficiency in terms of load balancing degree. The load balancing degree is considerably improved when throttle load balancing strategy is implemented at the broker level. Broker analysis the availability of VM and in case of unavailability, broker puts the cloudlet into the queue and cloudlet is not lost. As and when resource becomes available, resources are allotted to the job. Result in terms of throughput, load balancing degree and cost shows significant improvement ensuring reliability to be enhanced by the significant factor.

In future, we try to implement throttle load balancing strategy over the real cloud environment and analyse its performance in terms of energy consumption.

7. References

[1] J. Liu, S. Wang, A. Zhou, S. Kumar, F. Yang, and R. Buyya, “Using Proactive Fault-Tolerance Approach to Enhance Cloud Service Reliability,” *IEEE Trans. Cloud Comput.*, pp. 1–1, 2016.

[2] S. Veerapandi and K. Alagarsamy, “Fault Tolerance Avoidance in Cloud Computing Software Applications,” vol. 43, no. 3, pp. 166–169, 2017.

[3] A. Nathani, S. Chaudhary, and G. Somani, “Policy based resource allocation in IaaS cloud,” *Futur. Gener. Comput. Syst.*, vol. 28, no. 1, pp. 94–103, 2012.

[4] R. Buyya, R. Ranjan, and R. N. Calheiros, “InterCloud : Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services,” pp. 1–20.

[5] N. M. Gonzalez, T. Cristina, M. De Brito, and C. C. Miers, “Cloud resource management : towards efficient execution of large-scale scientific applications and workflows on complex infrastructures,” *IEEE Access*, 2017.

[6] N. Mangla and M. Singh, “Effect of Scheduling Policies on Resource Allocation in Market Oriented Grid,” in *2012 International Conference on Computing Sciences*, 2012, pp. 212–216.

[7] A. Zhou, Q. Sun, and J. Li, “Enhancing reliability via checkpointing in cloud computing systems,” *China Commun.*, vol. 14, no. 7, pp. 108–117, 2017.

[8] G. Z. Santoso *et al.*, “Dynamic Resource Selection in Cloud Service Broker,” in *2017 International Conference on High Performance Computing & Simulation (HPCS)*, 2017, pp. 233–235.

[9] J. G. “ Daeyong Jung, SungHo Chin, KwangSik Chung, HeonChang Yu1, “An Efficient Checkpointing Scheme Using Price History of Spot Instances in Cloud Computing Environment,” *IFIP Int. Fed. Inf. Process.*, pp. 185–200, 2011.

[10] X. Xu, L. Cao, and X. Wang, “Resource pre-allocation algorithms for low-energy task scheduling of cloud computing,” vol. 27, no. 2, pp. 457–469, 2016.

[11] S. Zhang, Z. Qian, Z. Luo, J. Wu, and S. Lu, “Burstiness-Aware Resource Reservation for Server Consolidation in Computing Clouds,” vol. 9219, no. c, pp. 1–14, 2015.

[12] G. Aupy, A. Benoit, and Y. Robert, “Energy-aware scheduling under reliability and makespan constraints,” *2012 19th Int. Conf. High Perform. Comput. HiPC 2012*, 2012.

[13] J. Liu *et al.*, “Using Proactive Fault - Tolerance Approach to Enhance Cloud Service Reliability,” pp. 1–13, 2016.

[14] I. Transactions, O. N. C. Design, and O. F. Integrated, “Reliability-Driven Energy-Efficient Task Scheduling for Multiprocessor Real-Time Systems,” vol. 30, no. 10, pp. 1569–1573, 2011.

[15] P. Pop, K. H. Poulsen, V. Izosimov, and P. Eles, “Scheduling and voltage scaling for energy/reliability trade-offs in fault-tolerant time-triggered embedded systems,” *Proc. 5th IEEE/ACM Int. Conf. Hardware/software codesign Syst. Synth. - CODES+ISSS ’07*, p. 233, 2007.

[16] A. Zhou *et al.*, “Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization,” *IEEE Trans. Serv. Comput.*, vol. XX, no. XX, pp. 1–1, 2016.

[17] G. Portaluri and S. Giordano, “A Power Efficient Genetic Algorithm for Resource Allocation in Cloud Computing Data Centers,” pp. 58–63, 2014.

[18] J. Choi, S. Member, B. Ahmed, R. Gutierrez-osuna, and S. Member, “Development and Evaluation of an Ambulatory Stress Monitor Based on Wearable Sensors,” vol. 16, no. 2, pp. 279–286, 2012.

[19] Y. Sharma, B. Javadi, W. Si, and D. Sun, “Journal of Network and Computer Applications Reliability and energy ef fi ciency in cloud computing systems : Survey and taxonomy,” vol. 74, pp. 66–85, 2016.

[20] T. J. Charity, “Resource Reliability using Fault Tolerance in Cloud Computing,” no. October, pp. 65–71, 2016.