International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 13 (2018) pp. 11189-11194
© Research India Publications. http://www.ripublication.com

Dynamic Placement with Virtual Machine Splitting for Fault Tolerance in Cloud Computing

Sunila
Student, Dept. Of Computer Science
Guru Nanak Dev University, Amritsar, Punjab, India.

Kamaljit Kaur
Assistant professor, Dept. Of Computer Science
Guru Nanak Dev University, Amritsar, Punjab, India.

Abstract

Cloud computing provides resources and remain significant part of modern computing. Cloud computing resources lacks as requirements corresponding to the clients increases. In such a situation VM management in cloud computing will required. This paper proposed a mechanism to manage resources within the cloud .Resource management utilized comparison form techniques. Client pass on the request for resources and if available resources are allotted .resources will be allotted simultaneously .This technique will be increase capacity of virtual management in cloud computing. In case deterioration does occur, VM migration strategy is in place. This will enhance the final production of the organization. Results are obtained in respect of their cost, and fault tolerance degree. Overall result shows improvement by 5%.

Keywords: Cloud computing, VM splitting, Resource allocation, load, virtual machine migration

INTRODUCTION

In cloud computing their Availability of resources requires efficient mechanism to balance the load and provide effective fault tolerance rate. The degree of fault tolerance increases in case fault tolerance strategy is in place.[1] The fault tolerance mechanisms which are commonly used are categorised into proactive and reactive fault tolerance. [2]both the mechanisms provide enhanced degree of fault tolerance. In proactive fault tolerance, fault is tackled before its occurrence hence it is based on the prediction. In reactive fault tolerance, fault is tackled when it enter into the system.

[3] In processing the system will comes in failure stage then is possible to recover and if the system is goes under current fault tolerance strategy then VM migration strategy must be in place to overcome the problems occurring due to load. [4]Systematic Error considered with in the Designed observation are ageing related faults. These faults causes the storage space to fail hence progress cannot be saved leading to degradation in terms of load and fault tolerance degree.

In Design phase contained [5] VM monitoring at load allocation phase. Auction based strategy is in place allowing the allocation of resources to the VM having sufficient resources available with them. This reduces the chance of fault at early stage of load allocation. In addition, VMs are capable of handling multiple jobs at the same time. [6]as the cloudlet allotted to the VM, resources tend to be utilized. Due to the aging affect, resources available with the VM cannot tackle the load even if it is within the threshold limit of the VM. In such a situation proposed system place efficient VM migration strategy consisting of sorted VM as stated by their resources.

The organisation of the paper is as under: section 2 presents literature survey that consists of strategies used to tackle load and fault tolerance. Section 3 gives the experimental setup, section 4 gives the proposed system, section 5 gives the performance analysis, section 6 gives the conclusion and future scope, section 7 gives the references

LITERATURE SURVEY

Temporary failure handling as proposed by [7] also known as Byzantine fault. Short transient faults are considered as Byzantine faults. These faults can be serious issue in concurrent fault handling mechanisms. To tackle these faults, combinational circuit was proposed that is capable of handling single Byzantine faults.

Energy conservation and fault tolerance proposed by [8]. Fault tolerance was accomplished by the use of check pointing. In case of failure, progress made by the task is saved and restarted from the position from where it is left off. Amount of checkpoints and check pointing placement is critical for conserving energy. Depending upon the amount of slacks and checkpoint overhead, energy can be conserved and result obtained through this system shows substantial improvement in terms of fault tolerance.

Hardware check pointing as described by [9] for archiving fault tolerance. Permanent hardware faults compensation along with internal states that must be tackled during recovery process was given priority. Last error Free State, represented through checkpoints and recovery procedure takes place from saved point hence saving power and achieving fault tolerance. Finite state machines were modelled to map software fault

tolerance technique to hardware fault tolerance through simple scan chain mechanism.

enhance pre-copy proposal as long as Live VM Migration proposed by [10]for decreasing downtime and migration time associated with transfer. Problem of pre-copy approach of delayed convergence is handled in this literature. The pre copy approach iterations become optimised by the use of bitmap. Frequently used pages were maintained within the bitmap. These pages were migrated to the destination node at the end of iterations and hence similar pages are not migrated again and again. This cause's downtime and migration time to reduce considerably.

Post copy approach of Live VM migration proposed by [11] for dynamic consolidation of virtual machines to optimize performance. The reactive approach for live VM migration in which extensive memory usage was present leads to performance degradation. Migration overhead was extensive in such situations .To recovered that situation post copy approach was proposed. The deteriorating machine was stopped at the source. The pages form the source was copied to destination machine and machine is started at destination to handle the overhead. Parallel transfer of pages to the destination node can be initiated by identifying the optimal server. Optimal server generally stores the images which can be recovered in case deteriorating virtual machine is detected.. Performance enhancement achieved through this literature in terms of downtime and migration time.

Online fault tolerance strategies studied through [12] to optimise performance was discussed through this literature. The challenges of proactive fault tolerance discussed through this literature was categorised in four distinct categories. 1. Aggressive fault managing 2. Flaw detection 3. Fault recognition and 4. Rallying procedures. Fault tolerance strategies implied on multicore/multiprocessor system for handling faults, comprehensively discussed through this literature. Energetic fault management described as discreet steps for block the failures of constituents in the order, since flaw detection is implementing to observe errors prior to conduct the way to implement the incompetent of the system. This literature further talk about fault diagnosis techniques that were used to track the unsuccessful components and verify the kind of a fault. Overly, recovery techniques just as checkpoint and service, reconfiguration and reallocation also elaborated through this literature.

Cluster Computing for low energy and power utilization proposed by [13] efficiently handle power and cooling requirements associated with virtual machines. High performance computing on two distinct domains was compared using this literature. A dedicated cluster compared against the Qemu virtual machines managing the cloud. Production and spark consumption furnish through both techniques were evaluated through ARM model. Variation of power consumption is observed in advanced system used within distributed computing. Advanced reservation mechanism is useful in the scenario where lack of resources lead to starvation. ARM system provides high end processing capabilities to tackle the needs of the users.

Resource provisioning under failure situations proposed by [14] for performance optimization. Public and private clouds merged together to form hybrid cloud and performance optimization mechanisms were incorporated into this system. Resource provisioning scheduling strategies employed in public and private clouds to handle failures. Time and area depending brokering strategies were employed to minimize cost and achieve Quality of service for applications executed on hybrid cloud.

Virtualization techniques employed as proposed by [15] to minimize downtime and migration time related with job execution in cloud. Time series based upgrade pre-copy application is used to tackle problems associated with downtime and migration time. Dirty pages in current execution environment were identified and placed at the end of iterations during copying phase.

Energy saving mechanism as proposed by [16] to identify high density I/O processes requiring check pointing and restart mechanism. CPU frequency and voltage during checkpointing operation if high consumes more energy. Through this literature, DVFS levels were minimized to conserve energy and achieving high fault tolerance rate.

Energy conservation as proposed by [17] through rollback/recovery mechanisms. Checkpoints were established at distinct time intervals. In case progress reaches those intervals progress was automatically saved. In case of failure, if progress was below the save point then rollback operation is applied to make the database at consistent state otherwise recovery procedure is performed to save the progress. Overall energy is conserved since in case of failure progress again begins from the state form where it is left off.

Energy conservation strategies were analysed by [18]. The investigation of energy conservation mechanisms implied on ICT machinery in this contemplates, focus was on energy regulation of the ICT setup breakup into two sections: Server as well as Network. It also place under software resources working on top of ICT equipment; take into Cloud System (CMS) province for organize a cloud infrastructure

Check pointing technique in time critical applications proposed by [19]. Failures increases as utilization of resources increases. To tackle the situation, progress is saved and in case of failure, application restarts from save point. Time and cost both optimized through the application of check pointing strategy. The cost effective energy efficient number of checkpoints to take single short lived fault was discovered. The checkpoints considered to be efficient, the total completion time (unmitigated execution time with re-execution time) to complete the work having minimum fault tolerance capacity.

Energy aware fault tolerance strategy proposed by [20] to optimize considered parameters. Firefly algorithm was incorporated to accomplish energy efficiency within the considered cloud system. Meta heuristic approach using firefly algorithm was used to support live VM migration. Distance between VMs considered as a parameter and nearest VM was selected for migration thus conserving energy and cost associated with migration.

Energy efficient approach proposed by '[21] in data comprehensive clusters was considered. An energy systematic flexible file replication system used to provide optimal energy conservation mechanism by transferring the progress in to the replicas in case main machine fails. Progress of the task saved was fetched again from backup and hence energy consumption decreases.

Task replication proposed by [22] for energy conservation. The proposed system deals with the methodology capable of providing energy efficient mechanism to allocate and tackle the jobs presented to datacentres. In order to do so, virtual machines are grouped in the form of high to low availability resources. In case deteriorated virtual machine is detected, recovery mechanism in terms of checkpoint is established.

Check pointing strategy proposed by [23] to tackle energy migration problems. Save point established through this literature helps in starting the work from where it is left off. Rather than large migration intervals, small migration intervals were used to make migration faster as compared to large migration intervals causing high downtime and migration time.

Prompted scheduling improvement strategy proposed by [24] through the check pointing in shared clusters for fault tolerance. Priority wise allocation of resources were considered. In case resource was allocated to least priority process and incoming process was of high priority then least priority process is prompted with the resource. The progress made by current process was saved and begun from same point from where it is left off. Response time considerably reduced through this literature.

Next section discusses the experimental setup used achieve the objective of the proposed system.

EXPERIMENTAL SETUP

Experimental setup described the software and parameters used in the proposed system. The simulation is conducted in net beans and cloudsim. The taxonomy used includes datacenters. Each datacenter comprises of 2 hosts and hosts are further partitioned into virtual machines. Brokers are declared corresponding to each host. These brokers are responsible for detecting the available resources with the VM for allocation. Bidding is done from the jobs which is accepted by broker and compared against the VM. In case bid is satisfied by the job, it is allotted to the VM.

PROPOSED SYSTEM

Proposed system minimise the resource consumption in order to save from deteriorating machine. Auction based strategy is accommodated along with VM migration strategy. Bids in terms of resource requirements are gathered by the broker and appropriate VM is selected after successful biding.

Within the overall system first of all we will detect the faults if faults will be occurred then this type of faults are named as overloaded faults. This faults will be decreases the overall functioning of the system. Within the proposed paper we will handle these faults by migrating their data from one VM to another VM so that it can be easily handled.

The methodology to be followed is given as under

Resource Provisioning: The selection, deployment of the resources that guarantee performance of the system as known as resources provisioning. The resource provisioning is done on the basis of requirement of cloudlet. Resource provisioning is on the base of processing element

For i=1: N

For j=1: N-i-1

If $(VM_j.PE > VM_{j_{+1}}.PE)$ *then*

Perform swapping of VM_j and VM_{j+1}

End of if

End of For

End of For

- If Resource_ Unavailable==true

  Performing scaling of Cloudlet requirements based on Bidding or auctions

  End of if

Scheduling: It specifies the way by which jobs execute in VM. It utilizes SJF technique to schedule the jobs in the VM. This will execute firstly the jobs that have minimum execution time.

- Shortest Job first scheduling mechanism is followed for job selection.
- Minimum=$Job.size$
- For i=2: Job. Size
- If(Minimum>$Job.size$)

  Minimum=Job_i

  End of if

  End of for

Monitoring: It will check the progress of the resource provisioning and scheduling by utilizing broker aware technique. It will check whether resources are available within the VM or not. Then allot the job to that VM.

- Broker Aware Monitoring is used
- $Broker = Job_i$ Broker receives the job
- $if\ Job_i.Resource \leq Vm_i.Resources$

  Allot the job to the VM

  I=i+1

  End of if

Since Job allocation is on the basis of capacity of the VM fault is avoided.

Fault detection

- Optimal VM selection can result in the selection of same VM again and again for task allocation. Leading to overheating and overflow faults.
- If(VM_i.Faults=True)
- Initiate Migration

VM Migration

- Broker aware migration process
- Broker checks for task requirement against the next available VM resource and migrate the task to next available VM without looking at maximum resources hence conserving time and cost

PERFORMANCE ANALYSIS AND RESULTS

The performance and result is observed in terms of energy consumed. For optimization this factor must be reduced. By following the above said approach energy consumption is significantly reduced. The energy consumption is given as under

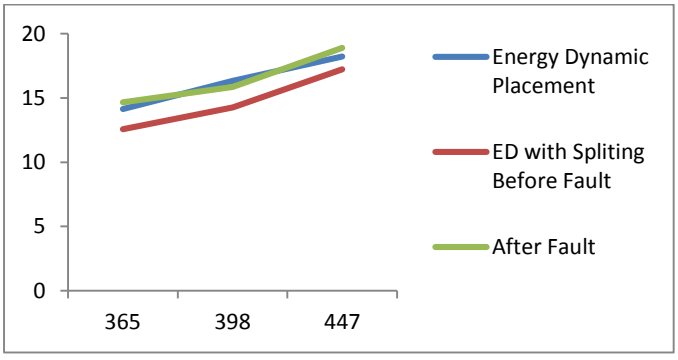
$$energy_{consumption} = \sum Energyconsumed_{idle} + Energyconsumedbyjob_i$$

The energy consumption is described as the total amount of energy ingest by the VM during performing a cloudlet and the total energy during idle time. Suppose the VM consumed energy at idle state is 14 joule and when cloudlet allocated to VM then energy is 300 then the total energy consumed will be 300+14=314

Table 1: Energy Consumption existing and proposed

Number of VMs	Energy(j)_Dynamic Placement	Dynamic Placement With Splitting	
		Before Fault	After Fault
365	14.153	12.565	14.67
398	16.33356	14.265	15.87
447	18.235	17.236	18.90

Energy consumption plots is given as under



Faults when injected in the form of selection of same VM for load allocation latency is introduced which is given as under

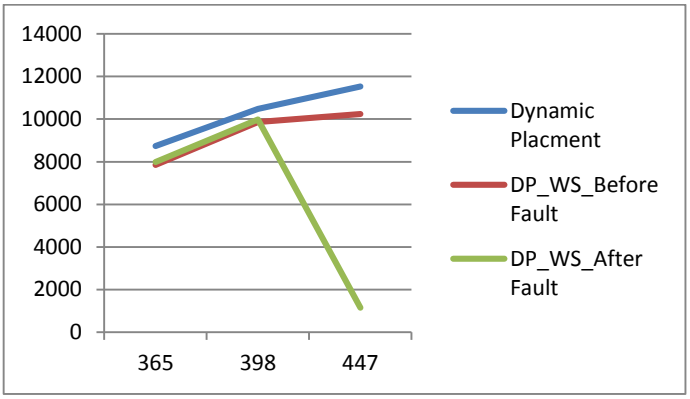
$$latency = Arrival_{time} - allocation_{time}$$

The latency is defined as the delay to allocate the resources to the cloudlets in VM. Suppose a cloudlet is arrived at time 3000ms and the resources allocated to that cloudlet at 3456ms then the total latency will be calculated as: 3456-3000=456ms.

Table 2: Latency associated with existing and proposed approach

Number of VMs	Dynamic Placement latency(ms)	Dynamic Placement with _Latency(ms)	
		Before Fault	After Fault
365	8730	7856	7986
398	10482	9865	9987
447	11526	10235	1145

Plots for the latency is given as under



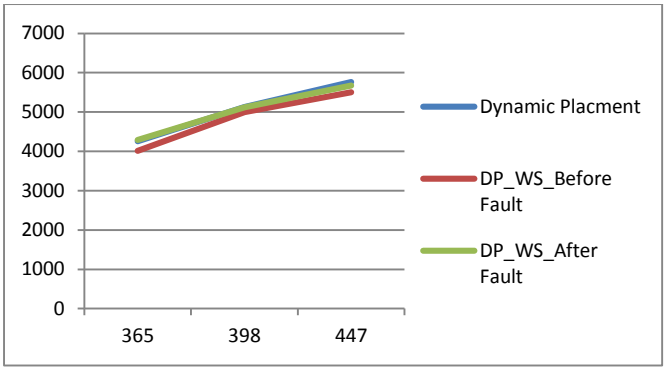
Performance analysis in terms of cost is given as under

$$cost = \sum vm_{cost} * cloudlet_i$$

Suppose 1 VM has encountered cost 20 for per cloudlet handling. Thus if a VM handles 300 cloudlets then the total cost encountered is 20*300=6000

Table 3: Cost associated with existing and proposed approach

Number of Cores	Dynamic Placement Cost	Dynamic Placement with Splitting	
		Before Fault	After Fault
365	4256	4012	4289
398	5126	5002	5123
447	5765	5501	5678



DP_WS means dynamic placement with splitting and overall performance metrics optimised by the use of proposed system.

CONCLUSION AND FUTURE SCOPE

Proposed system used bid based strategy in cloud to enhance performance of cloud system. Allocation of job in terms of best virtual machine selection is proposed. VM monitoring strategy is in place that continuously checks for the resource availability. VM availability ensures least latency. Faults can appear within the system and this literature considers faults occurring within the system due to single virtual machine selection again and again. VM migration strategy then comes into picture. Broker aware strategy for VM migration is used to enhance or optimise the latency, energy consumption and cost.

In future we can demonstrate this system against real cloud environment.

REFERENCES

[1] and F. Y. A. Zhou, S. Wang, Z. Zheng, C. Hsu, M. Lyu, "On cloud service reliability enhancement with optimal resource usage," *IEEE Trans. Cloud Comput.*, vol. 1, no. 1, pp. 1–1, 2014.s

[2] B. Mills, T. Znati, and R. Melhem, "Shadow Computing: An energy-aware fault tolerant computing model," *2014 Int. Conf. Comput. Netw. Commun.*, pp. 73–77, 2014.

[3] Y. Zhang, K. Chakrabarty, and S. Member, "A Unified Approach for Fault Tolerance and Dynamic Power Management in Fixed-Priority Real-Time Embedded Systems," vol. 25, no. 1, pp. 111–125, 2006.

[4] M. E. M. Diouri, O. Gl??ck, L. Lefevre, and F. Cappello, "Energy considerations in checkpointing and fault tolerance protocols," *Proc. Int. Conf. Dependable Syst. Networks*, pp. 3–8, 2012.

[5] X. Cui, B. Mills, T. Znati, and R. Melhem, "Shadow replication: An energy-aware, fault-tolerant computational model for green cloud computing," *IEEE ACCESS*, vol. 7, no. 8, pp. 5151–5176, 2014.

[6] M. Damodhar and S. Poojitha, "An Adaptive Fault Reduction Scheme to Provide Reliable Cloud Computing Environment," vol. 19, no. 4, pp. 64–73, 2017.

[7] A. S. F. Computing and H. A. Goosen, "The Byzantine Hardware Fault Model," vol. 8, no. 8930496, pp. 44–49, 1989.

[8] E. M. Elnozahy, "The Interplay of Power Management and Fault Recovery in Real-Time Systems," vol. 53, no. 2, pp. 217–231, 2004.

[9] D. Koch, C. Haubelt, and D.- Erlangen, "Efficient Hardware Checkpointing Concepts , Overhead Analysis , and Implementation," pp. 188–196, 2007.

[10] F. Ma, F. Liu, and Z. Liu, "Live Virtual Machine Migration based on Improved Pre-copy Approach," pp. 230–233, 2010.

[11] T. Hirofuchi, H. Nakada, S. Itoh, and S. Sekiguchi, "Reactive Consolidation of Virtual Machines Enabled by Postcopy Live Migration," pp. 11–18, 2011.

[12] H. Mushtaq, Z. Al-ars, and K. Bertels, "Survey of Fault Tolerance Techniques for Shared Memory Multicore / Multiprocessor Systems," pp. 12–17, 2011.

[13] K. L. Keville, R. Garg, D. J. Yates, K. Arya, and G. Cooperman, "Towards fault-tolerant energy-efficient high performance computing in the cloud," *Proc. - 2012 IEEE Int. Conf. Clust. Comput. Clust. 2012*, pp. 622–626, 2012.

[14] J. P. D. Comput, B. Javadi, J. Abawajy, and R. Buyya, "Failure-aware resource provisioning for hybrid Cloud infrastructure," *J. Parallel Distrib. Comput.*, vol. 72, no. 10, pp. 1318–1331, 2012.

[15] J. A. Johnson, "Optimization of migration downtime of virtual machines in," pp. 4–8, 2013.

[16] B. Mills, R. E. Grant, K. B. Ferreira, and R. Riesen, "Evaluating Energy Savings for Checkpoint / Restart," no. c, 2013.

[17] D. Ibtesham, D. Debonis, D. Arnold, and K. B. Ferreira, "Coarse-Grained energy modeling of rollback/recovery mechanisms," *Proc. - 44th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Networks, DSN 2014*, pp. 708–713, 2014.

[18] A. V Vasilakos, "Cloud Computing : Survey on Energy Efficiency," vol. 47, no. 2, 2014.

[19] A. Kumar, "An Efficient Ch heckpointing Approach h for Fault Tolerance in Tim me Critical Systems wi ith Energy Minimization," pp. 704–707, 2015.

[20] N. Jain and K. Inderveer, "Energy-aware Virtual Machine Migration for Cloud Computing - A Firefly Optimization Approach," *J. Grid Comput.*, 2016.

[21] Y. Lin, H. Shen, and S. Member, "EAFR : An Energy-Efficient Adaptive File Replication System in Data-Intensive Clusters Background of File Replication," vol. 13, no. 9, 2016.

[22] M. A. Haque and H. Aydin, "Energy-Aware Task Replication to Manage Reliability for Periodic Real-Time Applications on Multicore Platforms."

[23] T. Knauth, "VeCycle : Recycling VM Checkpoints for Faster Migrations," pp. 210–221.

[24] J. Li, C. Pu, Y. Chen, V. Talwar, and D. Milojicic, "Improving Preemptive Scheduling with Application-Transparent Checkpointing in Shared Clusters," pp. 222–234.