Identifying Overloaded Servers and Managing Dynamic Placement of Virtual machines in Cloud

Lamisha Rawshan
MS Student
Institute Information Technology,
University of Dhaka

Jobaer Islam Khan
Software Engineer
The Jaxara IT LTD,
Dhaka-1212, Bangladesh

Asif Imran
Lecturer
Institute of Information Technology,
University of Dhaka

ABSTRACT
Cloud computing is becoming one of the most popular commercial infrastructure due to its little maintenance expense and on demand resource utilization. Cloud computing possesses many kinds of technical challenges such as fault tolerance, reliability, availability, integrity etc. due to its complex and distributed nature. But the main problem related to all those is overload incurred by Virtual Machines (VM). So, load balancing is one of the most significant issues that can help to gain rapid performance of cloud infrastructure. This research proposes algorithms for detecting failed servers due to overloaded VMs. The failure detection algorithm checks server status after a predefined time interval. This algorithm gives proactive technique to deal with overloaded VMs. When any failure in the server is found, the resource balancing algorithm migrates its VMs to an adequate healthy Physical Machine (PM). To distribute workload evenly, the resource utilization skew is measured. This VM to PM mapping is done in a way that every PM will do almost equal amount of work.

General Terms
Cloud computing

Keywords
Cloud Computing, Resource management, Skewness, Virtual machine migration, Overload Detection.

1. INTRODUCTION
Cloud computing is an internet based computing that handles applications by sharing computing resources instead of local machines'. National Institute of Standards and Technology (NIST) defines cloud computing as a model that enables ubiquitous and useful network access to a shared group of computing resources based on their demand [1]. This group of shared resources can be allocated or released with little management effort or service provider interaction. According to IBM, cloud computing delivers computing resources based on user demand from data centers over the internet using pay-for-use basis. Cloud technology can improve resource utilization and reduce cost in many organizations [2]. It becomes important to guarantee that computing resources are properly utilized to meet application requirements [3].

Cloud is a group of physical machines pretending to be one computing environment [4].User see cloud as an illusion of unlimited computing resources [5]. But the main challenge is to manage the variability and heterogeneity of application requirement [6]. VMs shares the resources of PMs among several users to maximize the use of resources. The mapping of these PMs to VMs is very important to gain better performance of cloud computing [7]. This mapping is completely unknown to the users. They have no knowledge about the location of the PMs from where their VM is running. It is one of the main responsibilities of the cloud provider to meet the resource demand of users. The capability of PM must satisfy the resource demand of VMs running in it. Otherwise, the PMs will degrade its performance due to overloaded VMs [8].

In this research we present a methodology for detecting and avoiding possible failures due to overloaded VMs. Difference of average input rate and processing work rate for a certain period of time is used as the basis for measuring the load of users in a certain VM. When any overloaded server is detected, remapping from PMs to VMs is done in a way that the resource utilization skew of physical machine must be minimized. Skew is the measure of uneven resource utilization of physical machines. The proposed framework emphasizes on the remapping of physical machines to virtual machines when any overloaded PM is detected. This framework measures the difference between processing work rate and input rate of user to detect failure. Other researches [9], [10] have also measured processing rate, but they consider a block of streaming data. The proposed framework redistributes VM when overloaded VM is found, but existing research [9], [10] add fine or coarse grain of CPU adjustment.

The proposed model aims to detect failures from VMs of servers and to redistribute overloaded VMs to meet the resource demand in the cloud computing environment. Unevenly distributed load can cause degradation of overall performance in cloud computing. The proposed system also aims to give an almost equal amount of load in all physical machines. Failure detection leads to finding overload by VMs. It detects overload by comparing input jobs with processing jobs in a predefined time interval. More specifically, two main objectives of this research are-

1. Identification of slow servers running in the cloud to improve overall performance of application and meet user demand successfully.
2. Minimize resource utilization skew of PMs to reduce uneven resource distribution.

Empirical investigation of this method is performed in Ubuntu 14.04 Operating System(OS) and Openstack Juno cloud. MySQL database server will store historic data, recent usage data and servers capacity in different metrics that will be retrieved by fail detection, skew detection and resource balancing algorithms.

The remaining part of the document contains the following sections. Section 2 identifies related studies based on overload detection and dynamic allocation of virtual machines. Section 3 represents the architecture of the failure identification

component and mapping from VMs to PMs. Section 4 gives a detail description of proposed methodology and listed notations needed for algorithms. Section 5 formulates empirical results from the algorithms and analysis performance of this methodology. The last section that is section 6 concludes the research.

2. RELATED RESEARCHES

This section highlights recent works that are performed to detect failures and self-adaptation of cloud computing. An inaccurate resource allocation can cause insufficient use of resource, higher cost and low performance [11]. These researches have discussed about the significance of proper mapping from PM to VM.

A time based framework that monitors cloud for real time feedback is discussed in [12]. This framework gives self-healing mechanism when any failure is detected. The monitor collects cloud component behavior in binary output file and store in MySQL database. After a certain time monitor reads the database and detect the occurrence of undesirable activities. Then necessary actions are taken according to prespecified configuration. In this present research self-healing is ensured in a way that resource utilization skew must be minimized.

A high available middleware is used to tolerance fault in cloud based architecture [13]. It runs a deterministic algorithm that allocates replicated node when any failed node is found. It collects resource uses data from slave nodes. The algorithm sends notification to the system when resource usage of a node exceeds its predefined threshold. Then, this node is added to a queue N_{exceed}. The nodes of this queue are sorted in ascending order. This research replaces faulty nodes with additional nodes. But our present research migrates overloaded virtual machines to another physical machine.

Vijayakumar et al. [9] have presented a model to allocate resources dynamically for data streaming applications. The main objective of their research is to careful allocation of resources to avoid both over provision and the under provision. They detect buffer overflow by comparing time interval between receiving two blocks of streaming data and processing of one block of that data. When any overflow occurs their model increase amount of CPU based on requirement. They define a fixed amount of CPU percentage for multiplicative increase and additive decrease of CPU. They evaluate their model in both static and dynamic environments. They use a fixed amount of CPU addition which is costly.

Wang et al. [14] have proposed a fail detection model for web applications running in the cloud. Their proposed method is based on the assumption that sudden increase and decrease of workload can change behavior of tradition web applications. The method of this paper characterizes customers' workload from their access behavior vector and volume vector. One of their objective was to group users with similar workload to recognize access behavior pattern as they have similar access behavior matrix and resource demand. They use an online clustering method for recognizing access behavior pattern of users from the collected vectors. Canonical Correlation Analysis (CCA) is used between workload of users and performance matrix of application to detect failure. Fault detection accuracy of this model fully depends on the clustering method.

Sampaio et al. [5] targets to improve cloud infrastructure power efficiency using dynamic algorithms which map physical machines to virtual machines. They use a proactive fault tolerance technique to identify failures in systems. This model contains a cluster coordinator to manage cloud users' submitted job. Each submitted job is seen as a set of independent job and a time limit that is the deadline is added to each job. Cluster coordinator decides which VM will be mapped to which PM.

Sadeka et al. [15] have proposed a method that will predict future resource demand and resource provision strategy in cloud computing. They uses historical data and recent resource uses to predict future throughput with machine learning algorithms. They evaluate their proposed model with mean Absolute Percentage Error, Root Mean Squared error etc.

Aameek et al.have presented agile data center with integrated and virtualization technology [16]. Their research doesn't add even distribution of workload.

Rayhan et at. have proposed a resource provisioning scheme for cloud computing with distributed decision maker [17]. Individual physical machines take their own resource provisioning decision. Multi Attribute Utility Theory is used for VM allocation and migration. VM allocation and migration decision is affected by many attribute like resource availability, network bandwidth, network cost, Service Level Objectives (SLO) violation. In this scheme, nodes are organized using unstructured P2P architecture to avoid single point of failure threat. Every physical machine collects information from its neighbored with a constant distance.

Load balancing algorithms are used to balance load on the whole system in order to meet the objectives and to generate improved results [18]. Some of the challenges of load balancing are- virtual machines migration, stored data management, energy management etc. [19]. Mauro et al. give algorithms for dynamic load management and reallocation of virtual machines in cloud [11]. They focus on three main steps of virtual machine migration process- when to migrate distributed virtual machines, which virtual machine is suitable for migration, the process to reallocate virtual machines to physical machines. Their proposed algorithm is based on four phases. The first phase is a selection of the sender host by using Cumulative Sum (CUSUM) algorithm. The second phase is the selection of guest host to decide which of the guest should migrate. The third phase is a selection of receiver host to migrate the guest host. The last phase is assignment of the guest host.

Su-Ching et al. have introduced a load balancing framework with a two-level scheduling algorithm [20]. This algorithm uses a mixed of opportunistic load balancing and load balance min-min algorithms. Here, an agent collects information and store data. This data are compared with a predefined threshold.

All these works give algorithms for detecting faulty processes and provisioning resource based on clients demand. Maximum of them adds resources to an existing resource or replicate faulty resource with new one. On the other hand our algorithm output resource reallocation map to migrate failed VMs to another PM that are capable to continue normal work. This migration will be done in a way that all PM will do almost equal amount of work.

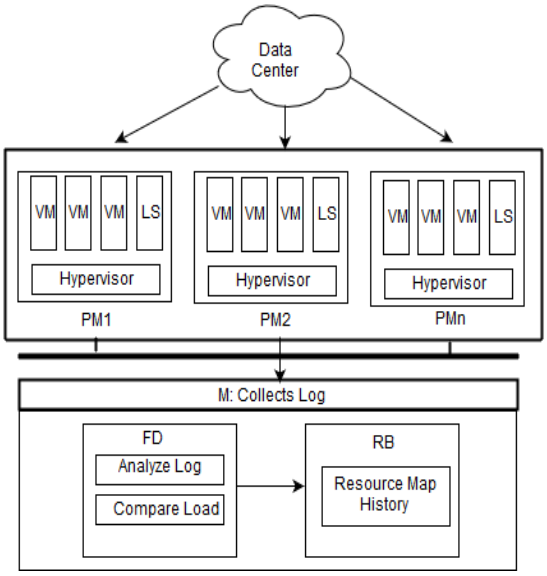


Fig 1: Architecture of proposed system

3. SYSTEM ARCHITECTURE

This section provides a detail description of the proposed architecture. The architecture is a hybrid cloud environment which contains both centralize and distributed resources. It contains a set of homogeneous physical machines, P= {PM_1, PM_2,…., PM_N}. Each PMs can allocate many Virtual Machines (VM). Virtual machine instances are created based on users demand. These machines contain a mix of database servers, web servers, mail servers etc. Local Statistics (LS) collects logs of individual machine. The architecture is depicted in Figure 1.

The proposed system contains two main parts- Failure Detector (FD) and Resource Balancer (RB) that amalgamates some other entity. This model contains a Monitor (M). M collects log from local statistics of individual machines after a certain period. The resource balancer remap VMs to PMs when any failure is found. The entities of this architecture are described in this section.

- Monitor (M) - Every PM runs a hypervisor, which supports multiple applications. M collects each machines statistics from LS and store collected data in information schema after a small time interval. It passes collected information to fail detector.
- Failure Detection (FD) - FD determines the chances of failure by using metadata of information schema. FD becomes alive after a time period t and detects the overloaded servers.
- Resource Balancer (RB) - FD passes information of overloaded server RB. It calculates skew for every PM and sort in decreasing order. RB select PM that have low skew and needed space for overloaded VM. It outputs new map of VMs to PMs. This new map indicate which VM will be migrated to which PM.

4. PROPOSED METHODOLOGY

This section describes the algorithm for detecting failure and necessary self-healing to recover from failed condition. The

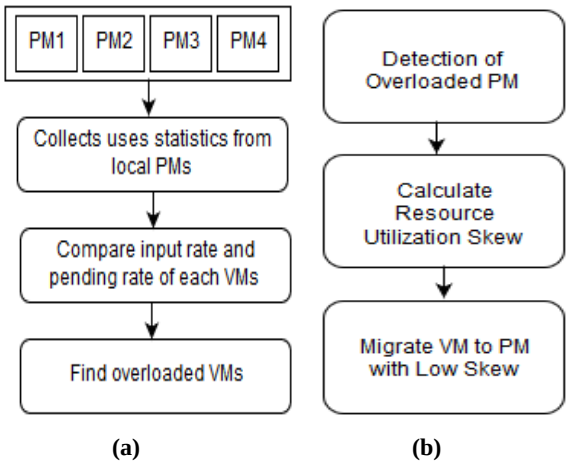


Fig 2: Flow charts for load detection and resource allocation

algorithm maintains a timer, $SpareTime$ after which it becomes alive to collect data. It collects Input Rate, Output Rate, $MaxMemVector$, and $ResourceUtilizationVector$ over a specified interval. The $FailureDetector$ algorithm detects overloaded server by comparing the average input rate and processing rate of that server in the cloud. To reduce the effect of failure, self-adaptation mechanism is needed. Self-adaptation consists of $SkewDetection$ and $ResourceMapGenerator$ algorithms. Flow chats of load detection and resource allocation is depicted in figure 2. The $SkewDetection$ algorithm detects the percentage of unevenness of resource utilization among the PMs. It also detects resource utilization skew of servers within a PM by combining resources utilization of its VM. If skew is presented

1: $input$= N, MaxMemVector, ResourceUtilizationVector

2: $Output$= OverloadedPMsIDVector

3: **procedure** FAILURE DETECTION

4: **for** i ←1 to N **do**

5: $m \leftarrow Number\ of\ VMs\ running\ on\ i^{th}\ PMs$

6: **for** j ←1 to m **do**

7: $AvgInput \leftarrow Average\ input\ rate\ of\ j^{th}\ VM\ instance$

8: $AvgProcessing \leftarrow Average\ processing\ rate\ of\ j^{th}\ VM\ instance$

9: **if** $AvgPending > AvgInput$ **then**

10: $OverloadedPMsIDVector \leftarrow i$

11: **end if**

12: **end for**

13: return $OverloadedPMsIDVector$

14: **end for**

15: **end procedure**

Fig 3: Algorithm for failure detection

$nput$= N, ResourceUtilizationVector

2: $Output$= $Skew[N]$

3: **procedure** SKEWDETECTION

4: **for** i ←1 to N **do**

5: $m \leftarrow Number\ of\ VMs\ running\ on\ i^{th}\ PMs$

6: **for** j ←1 to m **do**

7: $Deviation \leftarrow \sqrt{\sum_{1}^{j=N} \frac{(U_i - U_{avg})^2}{N}}$

8: **end for**

9: $S \leftarrow \frac{D}{U_{avg}}$

10: $Skew[i] \leftarrow S$

11: **end for**

12: return $Skew[N]$

12: **end procedure**

Fig 4: Algorithm for skew detection

1: $input$= $Skew[N]$, ResourceUtilizationVector, OverloadedPMsIDVector

2: $Output$= $ResourceReAllocationMap$

3: **procedure** RESOURCEALLOCATION

4: **for** i ←1 to N **do**

5: $m \leftarrow Number\ of\ VMs\ running\ on\ i^{th}\ PMs$

6: **for** j ←1 to m **do**

7: $ServerID \leftarrow AllocateResourse(Skew[i], ResourceUtilization[i][j], OverloadedPMsIDVector)$

8: $ResourceReAllocationMap \leftarrow ResourceMap[ServerID][j]$

9: **end for**

10: **end for**

11: return $ResourceReAllocationMap$

12: **end procedure**

Fig 5: Algorithm for resource allocation

in the cloud, overloaded server will be redistributed among nodes with minimum skew. If skew is not present that is memory consumption is uniform across nodes, deallocation of unnecessary applications will be happened to fulfill requirement of overloaded servers. $ResourceMapGenerator$ uses the skew value of PMs and generate a resource map to reallocate VMs to PMs.

4.1 Notation and Preliminaries

This subsection presents needed symbols for describing algorithms. Let,

$SpareTime$= A time factor after which the monitor becomes alive.

$InputRate$= Input rate of a VM instance in the $SpareTime$.

$OutputRate$=Processing rate of a VM instance in the $SpareTime$.

$LocalLoad$= Cumulative memory in use by all VM instance of a node.

$MaximumMemory$= Maximum memory of a node.

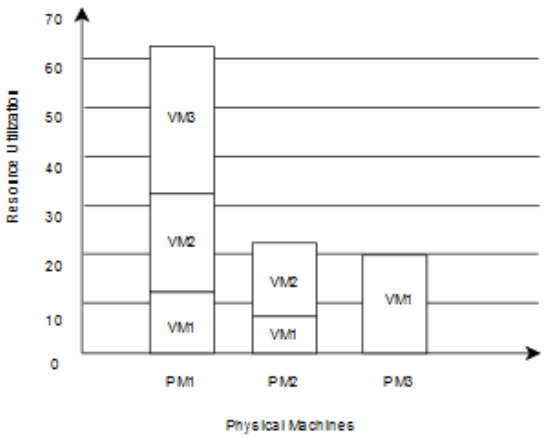
$ResourceUtilizationVector$= Vector which contains a list of memory consumption of all nodes. <$LocalLoad_1$, $LocalLoad_2$, $LocalLoad_3$,……., $LocalLoad_N$>, where the total number of nodes is N.

$MaxMemVector$= Vector which contains a list of MaxMemVector of all nodes. <$MaximumMemory_1$, $MaximumMemory_2$,$MaximumMemory_3$,..., $MaximumMemory_M$>, where the total number of nodes is M.

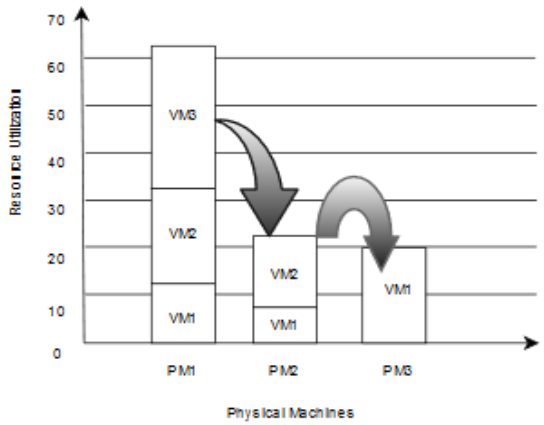
N= Number of Physical Machine

4.2 Failure Detection

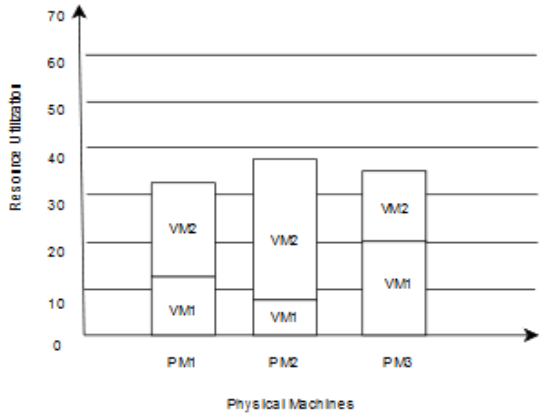
This research uses active monitoring load balancer to detect overload. The challenges to develop this kind of algorithm is to estimation and comparison of load, system stability and performance etc. [21]. Failure Detection algorithm takes 3 inputs- Number of physical machine in whole cloud, $MaxMemVector$ and ResourceUtilizationVector. It calculates the average arrival rate of input data of a server and



(a)



(b)



(c)

Fig 6: Migration of load using skewness

the average processing rate of data. It compares these two values. If the average arrival rate of input is greater than the average processing rate of that input in a specific server, it means that data is arriving much faster than the current node capability. If the average processing rate of data is greater than the normal processing rate, resource deallocation is possible from this server. Algorithm 1 is given in figure 3.

4.3 Self-Adaptation

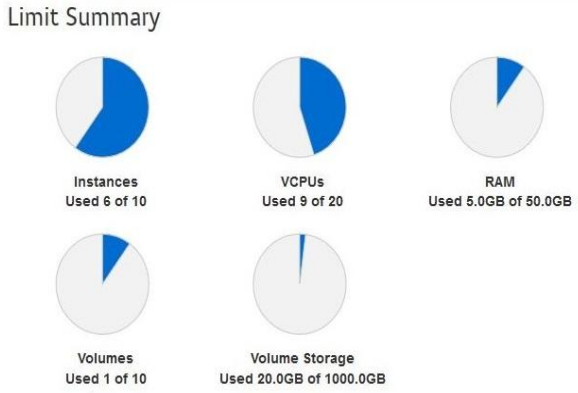


Fig 7: Resource Utilization of cloud

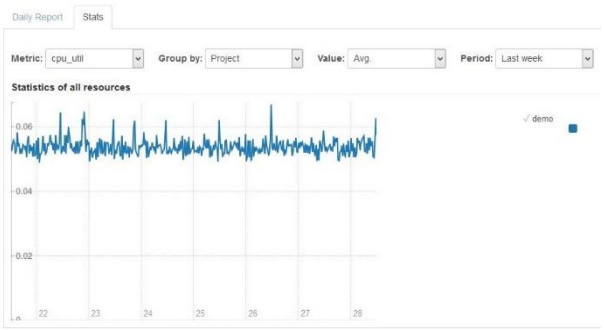


Fig 8: CPU utilization

4.3.1 Skew Detection

When any overloaded server is detected self-adaptation algorithm tries to allocate needed space immediately. One common property of cloud computing that can cause low resource distribution detection of skew is more or less evenly distribution of resources. So it is necessary to distribute resources in such a way that all servers of a node will consume approximately similar amount of memory. In order to prevent unevenness of little skew and enough space will be chosen for migration of an overloaded server. Equation (1) and (2) contain formulas for detecting skew of physical machines.

$$\text{Deviation, } D = \sqrt{\sum_{1}^{i=N} \frac{(U_i - U_{avg})^2}{N}} \quad (1)$$

U_i, indicates memory uses of i^{th}server in the spare time. N is the total number of resources running in a PM. The value of D will generate deviation from the average used recourse in a server. At last PM will generate the skew of the server. From the value of PM resource redistribution decision will be made.

$$\text{Skew, } S = \frac{D}{U_{avg}} \quad (2)$$

This algorithm is presented in Figure 4. The $SkewDetection$ algorithm takes 2 inputs- Number of physical machines and Resource utilization metrics. By using equation 1 and 2 it determines the deviation from average usage for every server. It calculates Skew for every PM.

4.3.2 Resource Allocation

$ResourceMapGenerator$ algorithm (Figure 5) determines which PM can continue normal work flow of overloaded servers. The input of this algorithm is -Skew array of all PMs, overloaded servers and resource utilization details. It sort this Skew list in decreasing order. The main purpose is to choose a PM with low skew and enough space for migrating overloaded server.

5. RESULT

This research has used OpenStack to check the performance of the proposed algorithm. OpenStack provides a platform for managing large amount of virtual machines. It is an open system so open-source tools can be easily applied here. The implementation contains real life software running on the virtual instance of OpenStack. Here VM instances are used as resource. Migration of load from overloaded PMs is given in Figure 6. The algorithm is applied to a dashboard. The whole system is monitored using the dashboard after a time interval for detecting any failures due to overloaded VMs. Resource utilization skew is measured using the formula discussed above. VMs are distributed in way that skew will be minimized. The usage summary of overall cloud is depicted in Figure 7. Figure 8 represents the CPU utilization of cloud.

6. CONCLUSION

This paper have proposed a methodology to detect overloaded servers which are running in multiple PMs. When any overloaded server is found, it dynamically maps VMs to PMs. The main goal of this work is to avoid failures due to overload by users, which implies to increase performance of a hybrid cloud computing infrastructure. To achieve this goal, three algorithms are developed- Failure Detection, Skew Detection and Resource Allocation. These algorithms give overloaded servers enough resources to continue their running task without any interruption. Resource are distributed in a way that resource utilization skew of every physical machine will be minimized. Which improves overall working efficiency of the cloud.

7. ACKNOWLEDGMENTS

A sincere thanks go to the reviewers of this paper for their valuable review.

8. REFERENCES

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," 2011.

[2] Imran, Asif, Alim Ul Gias, and Kazi Sakib. "An empirical investigation of cost-resource optimization for running real-life applications in open source cloud." *High Performance Computing and Simulation (HPCS), 2012 International Conference on*. IEEE, 2012.

[3] Beloglazov, Anton, and Rajkumar Buyya. "Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints." *Parallel and Distributed Systems, IEEE Transactions on* 24.7 (2013): 1366-1379.

[4] Barrett, Diane. "Security Architecture and Forensic Awareness in Virtualized Environments." Cybercrime and Cloud Forensics: Applications for Investigation Processes: Applications for Investigation Processes (2012): 129.

[5] Sampaio, Altino M., and Jorge G. Barbosa. "Dynamic power-and failure-aware cloud resources allocation for sets of independent tasks." *Cloud Engineering (IC2E), 2013 IEEE International Conference on*. IEEE, 2013.

[6] Gong, Zhenhuan, et al. "Siglm: Signature-driven load management for cloud computing infrastructures." *Quality of Service, 2009. IWQoS. 17th International Workshop on*. IEEE, 2009.

[7] Kim, Hongjae, et al. "A Novel Adaptive Virtual Machine Deployment Algorithm for Cloud Computing." proceedings of International Conference on Information Science and Industrial Applications (ISI 2012), Philippines. 2012.

[8] Prathima, S., and Shaik Shasha Ali. "Dynamic Resource Allocation using Virtual Machines for Cloud Computing Environment."

[9] Vijayakumar, Smita, Qian Zhu, and Gagan Agrawal. "Dynamic resource provisioning for data streaming applications in a cloud environment." *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*. IEEE, 2010.

[10] Cao, Junwei, and Wen Zhang. "Dynamic Controlling of Data Streaming Applications for Cloud Computing."

[11] Arias, Michael, et al. "A Framework for Recommending Resource Allocation based on Process Mining."

[12] Rawshan, Lamisha, Kazi Sakib, and Asif Imran. "Time-Waved Monitoring and Emergent Self Adaption of Software Components in Open Source Cloud." *Proceedings of the The International Conference on Engineering & MIS 2015*. ACM, 2015.

[13] Imran, Ali, et al. "Cloud-Niagara: A high availability and low overhead fault tolerance middleware for the cloud." *Computer and Information Technology (ICCIT), 2013 16th International Conference on*. IEEE, 2014.

[14] Wang, Tao, et al. "Fault detection for cloud computing systems with correlation analysis." *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*. IEEE, 2015.

[15] Islam, Sadeka, et al. "Empirical prediction models for adaptive resource provisioning in the cloud." *Future Generation Computer Systems* 28.1 (2012): 155-162.

[16] Singh, Aameek, Madhukar Korupolu, and Dushmanta Mohapatra. "Server-storage virtualization: integration and load balancing in data centers." *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. IEEE Press, 2008.

[17] Rahman, Raziur, et al. "A peer to peer resource provisioning scheme for cloud computing environment using multi attribute utility theory." *Innovative Computing Technology (INTECH), 2013 Third International Conference on*. IEEE, 2013.

[18] Khiyaita, A., et al. "Load balancing cloud computing: state of art." *Network Security and Systems (JNS2), 2012 National Days of*. IEEE, 2012.

[19] Rashmi, K. S., V. Suma, and M. Vaidehi. "Enhanced load balancing approach to avoid deadlocks in cloud." *arXiv preprint arXiv:1209.6470* (2012).

[20] Wang, Shu-Ching, et al. "Towards a load balancing in a three-level cloud computing network." *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*. Vol. 1. IEEE, 2010.

[21] Bhaskar, R., S. R. Deepu, and B. S. Shylaja. "Dynamic allocation method for efficient load balancing in virtual machines for cloud computing environment." *Advanced Computing* 3.5 (2012): 53.