ISSN No. 0976-5697

Volume 5, No. 3, March-April 2014

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Server Consolidation Algorithms for Virtualized Cloud Environment with Variable Workloads: A Performance Evaluation

Susheel Thakur*
Department of Computer Science
Himachal Pradesh University
Shimla, India

Arvind Kalia
Department of Computer Science
Himachal Pradesh University
Shimla, India

Jawahar Thakur
Department of Computer Science
Himachal Pradesh University
Shimla, India

Abstract: Server Consolidation is an efficient approach towards the utilization of physical machines, in order to reduce the total number of servers that an organization requires. To prevent the server sprawl, this practice was developed. Server Consolidation allow large-scale data centers to improve their resource utilization and energy efficiency using virtualization technologies. To prevent server sprawl, server consolidation aims at reducing the number of server machines used in the data centers by consolidating load and enhancing resource utilization of physical systems. Virtualization enables the migration of virtual machines (VMs) between the physical machines using the technique of live migration mainly for improving the efficiency. Virtual machine migration is promising approach to realize the objectives of efficient, adaptive and dynamic resource management in virtualized cloud environment. In this paper, a comprehensive study of the server consolidation algorithms and their usage towards dynamic resource management in the virtualized cloud environment is presented. We try to simulate and investigate the impacts of different server consolidation algorithms on the performance of the live machine migration in both source and target machine in terms of response time. Here in this paper, a performance evaluation of the chosen heuristics and fundamental insights obtained when variable load is generated over the physical machines, aimed at reducing server sprawl, optimizing power consumption and load balancing across the physical machines in virtualized cloud environment is presented.

Keywords: Cloud computing; live migration; Load balancing; Server Sprawl; Virtual Machine Monitor (VMM), Hotspot mitigation.

I. INTRODUCTION

In 1969, Leonard Kleinrock [15], one of the chief scientists of the original Advanced Research Projects Agency Network (ARPANET) which seeded the Internet, said: "As of now, computer networks are still in their infancy, but as they grow up and become sophisticated, we will probably see the spread of computer utilities which, like present electric and telephone utilities, will service individual homes and offices across the country." This vision of computing utilities based on a service provisioning model anticipated the massive transformation of the entire computing industry in the 21st century whereby computing services will be readily available on demand, like other utility services available in today's society. Cloud Computing is defined by NIST[21] as a model for enabling convenient, on demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction.

For simplicity, a cloud is a pool of physical computing resources i.e. a set of hardware, processors, memory, storage, networks, etc. which can be provisioned on demand into services that can grow or shrink in real-time scenario[27]. Virtualization plays a vital role for managing and coordinating the access from the resource pool. A virtualized environment that enables the configuration of systems (i.e. compute power, bandwidth and storage) as well as the creation of individual virtual machines is the key features of the cloud computing. Virtualization is ideal for delivering cloud services. Virtualization Technology enables the decoupling of the application payload from the underlying physical hardware and provides virtualized resources for higher-level applications. An important feature of a virtual machine is that software running inside it is limited to resources and abstractions provided by the VM. The software layer that provides the virtualization is called virtual machine monitor (VMM). VMM virtualizes all of the resources of physical machine, thereby supporting the execution of multiple virtual machines. Virtualization can provide remarkable benefits in cloud computing by enabling VM migration to balance load across the data centers [13].

In the surge of rapid usage of virtualization, migration procedure has been enhanced due to the advantages of live migration say server consolidation and resource isolation. Live migration of virtual machines [5] [18] is a technique in which the virtual machine seems to be active and give responses to end users all time during migration process. Live migration facilitates energy efficiency, online maintenance and load balancing [14]. Live migration helps to optimize the efficient utilization of available CPU resources.

Server consolidation is an approach to the efficient usage of computer server resources in order to reduce the total number of servers or server locations that an organization requires. This approach was developed in response to the problem of "server sprawl". Server sprawl is a situation in which multiple underutilized servers

accommodate more space and consume more resources that can be justified by their workload. Many organizations are turning to server consolidation to reduce infrastructure complexity, improve system availability and save money. With increasingly powerful computing hardware, including multi-core servers; organizations can run large workloads and more applications on few servers. Reducing the numbers of servers has tangible benefits for the data centers as well.

Consolidation will result in reduced power consumption and thus reducing overall operational costs for data center administrators. Live migrations achieve this. Based on the load conditions, under-utilized machines having resource usage above a certain threshold are identified and migrations are triggered to tightly pack VMs to increase overall resource usage on all PMs and free up resources/PMs if possible[3].

The rest of this paper is organized as follows: Section II describes the survey of existing literature of various server consolidation algorithms for cloud computing environment. Section III provides the overview of the chosen server consolidation algorithms for performance analysis in virtualized cloud environment. Section IV gives details about the experimental test bed used in performance analysis. Section V discusses the experimental evaluation and results. Section VI provides the conclusion and future work.

II. RELATED WORK

Several research groups are working on server consolidation in both academia and industry. This section presents studies and systems related to server consolidation. Khanna et al. [11] presented a dynamic management algorithm, which is triggered when a physical server machine becomes overloaded or underloaded. The main aim of their algorithm was to: i) guarantee that SLAs are not violated (SLAs are specified in terms of response time and throughput); ii) minimizing the migration cost; iii) optimizing the residual capacity of the system; and iv) minimizing the number of physical machines used. Bobroff et al. [19] proposed a dynamic server consolidation algorithm to reduce the amount the amount of required capacity and the rate of SLA violations. This heuristics make use of historical data to forecast future demand and relies on periodic executions to minimize the number of physical machines to support the virtual machines. Mehta and Neogi [23] developed the ReCon tool, aimed at recommending dynamic server consolidation in multi-cluster data centers. ReCon takes into account both static and dynamic costs of physical machines, the costs of VM migration and the historical resource consumption data from the existing environment in order to provide an optimal plan of VMs to physical machine mapping over time.

Wood et al. [26] presented the sandpiper system for monitoring and detecting hotspots, and remapping VMs whenever necessary. In order to choose which VMs to migrate, Sandpiper sorts them using a volume-to-size (vsr) metric, which is a based on cpu, network, and memory loads. Sandpiper migrates the most loaded VMs from an overloaded physical machine to one with sufficient capacity. Srikantaiah et al. [24] studied the problem of request scheduling for multi-tier web applications in virtualized heterogeneous systems, to minimize the energy consumption, while trying to meet the performance requirements. They try to investigate the effects of performance degradation due to high utilization of different resources when the workload is consolidated. They have determined that the energy consumption per transaction results in a "U"-Shaped curve, and it is possible to find the optimal utilization point. The authors have developed a heuristic for the multidimensional bin packing problem as an algorithm for the consolidation of workload to handle the optimization over multiple resources. However, this approach is workload type and application dependent.

Cardosa et al. [17] have developed an approach for the problem of power-efficient allocation of VMs in virtualized heterogeneous computing environments. They have used the min, max and shares parameters of Xen's VMM, which represents minimum, maximum and proportion of the CPU allocated to VMs sharing the same resource. However, the approach suits only to enterprise environments as it does not support strict SLAs and requires the knowledge of application priorities to define the shares parameter. Speitkamp and Bichler [4] [16] introduced a linear programming for the static and dynamic server consolidation problems. They also formulated extension constraints for limiting the number of virtual machines in a physical server, mapping virtual machines to a specific set of physical servers that have some unique attribute and, limiting the total number of migrations for dynamic consolidation. In addition, they also designed an LP-relaxation based heuristic for minimizing the cost of solving the linear programming formulations. Emmanuel et al. [6] proposed a dynamic resource allocation framework based on their load balancing VM migration algorithm on a test-bed of three ESX servers, a SAN, VMware VC (Virtual Center) and VIBM Migration Handler. Similarly, Verma et al. [1] presented the pMapper architecture and a set of server consolidation algorithms for heterogeneous virtualized resources. The algorithms take into consideration power and migration costs and the performance benefit when consolidating applications into physical servers.

Ameek et al. [2] has developed a HARMONY set-up with ESX server and SAN (storage area network) controller for integrating both server and storage virtualization technologies to design an agile data centers. Keller et al. [10] design Golondrina multi-resource management for operating system-level virtualized environment with client systems, manager server and cluster gate. Starling [12] introduced affinity based VM placement and migration in a decentralized approach with an 8-node cluster of 2x dual-core AMD machines. Jung et al. [8] [9] have investigated the problem of dynamic consolidation of VMs running a multi-tier web-application using live migration, while meeting SLA requirements. The SLA requirements are modelled as the response time precomputed for each type of transactions specific to the web-application. A new VM placement is produced using bin packing and gradient search techniques. The migration controller decides whether there is a reconfiguration that is effective according to the utility function that accounts for the SLA fulfilment. However, this approach can be applied only to a single web-application setup and, therefore, cannot be utilized for a multitenant IaaS environment.

III. SERVER CONSOLIDATION ALGORITHMS

Server consolidation is an approach to the efficient usage of computer server resources in order to reduce the total number of servers or server locations that an organization requires. This approach was developed in response to the problem of "server sprawl". In order to reduce server sprawl in the data centers, server consolidation algorithms are implemented. These algorithms are VM packing heuristics which try to pack as many VMs as possible on the physical machine (PM) so that resource usage is improved and under-utilized machines can be turned off.

A. Sandpiper:

Sandpiper is a system that automates the task of monitoring and detecting hotspots, determining a new mapping of physical resources to virtual resources, by resizing or migrating VM's to eliminate the hotspots. Sandpiper makes use of automated black-box and gray box strategies for virtual machine provisioning in cloud data centers. Specifically the black-box strategy can make decisions by simply observing each virtual machine from the outside and without any knowledge of the application resident within each VM. The authors present a gray-box approach that assumes access to OS-level statistics in addition to external observations to better inform the provisioning algorithm. Sandpiper implements a hotspot detection algorithm that determines when to resize or migrate virtual machines, and a hotspot migration algorithm that determines what and where to migrate and how many resources to allocate. The hotspot detection component employs a monitoring and profiling engine that gathers usage statistics on various virtual and physical servers and constructs profiles of resource usage. These profiles are used in conjunction with prediction techniques to detect hotspots in the system. Upon detection, Sandpiper grants additional resources to overloaded servers if available. If necessary, Sandpiper's migration is invoked for further hotspot mitigation. The migration manager employs provisioning techniques to determine the resource needs of overloaded VMs to underloaded servers.

Sandpiper supports both black-box and gray-box monitoring techniques that are combined with profile generation tools to detect hotspots and predict VM Resource requirements. Hotspots are detected when CPU usage values are violated with respect to the CPU thresholds set. Physical machines (PMs) are classified as underloaded or overloaded. The PMs are sorted based on the descending order of their volume metric, and VMs are sorted based on the descending order of their vsr metric, where volume and vsr are computed as:

$$vol = \left(\frac{1}{1-cpu}\right) * \left(\frac{1}{1-mem}\right) * \left(\frac{1}{1-net}\right) \quad (1)$$

$$vsr = \frac{vol}{size} \quad (2)$$

where cpu, memory and n/w refers to cpu, memory and n/w usages of the PMs and VMs respectively and size refers to the memory footprint of the VM.

To mitigate hotspot on an overloaded PM, the highest vsr VM is migrated to a least loaded PM amongst the underloaded ones. If the least loaded PM can't house the PM, next PM in the sorted order is checked. Similarly, if the VM cannot be housed in any of the underloaded PMs, next VM in the sorted order is checked. This way sandpiper tries to eliminate hotspots by remapping VMs on PMs through migration. The experimental results showed that migration overhead is less than that of swapping overhead; however, swapping increases the chances of mitigating hotspots in cluster with high average utilization [25] [26].

B. Khanna's Algorithm:

Khanna et al., in [11] [25], proposed Dynamic Management Algorithm (DMA) that is based on Polynomial-Time Approximation Scheme (PTAS) heuristic algorithm. The algorithm operates by maintaining two types of ordering lists, which are migration cost list and residual capacity list. The PMs are sorted according to the increasing order of their residual capacities across any resource dimension like CPU. The VMs on each PM are sorted according to the increasing order of their resource utilization like CPU usage. Migration costs of the VMs are determined based on their resource usage i.e. high usage implies high costly migration. Whenever a hotspot is detected on a PM due to violation of upper threshold, VM with least resource usage is chosen for migration to target host which has the least residual capacity to house it. If a PM cannot accommodate the VM, next PM in the sorted order is checked. Similarly, if the VM cannot be accommodated by any of the candidate target PMs, next least usage VM from the sorted order is checked.

Whenever coldspots are detected, the least usage VMs across all the underloaded PMs is chosen and migrated to a targeted PM, only if addition of the new VM increases the variance of residual capacities across all the PMs, else we choose the next VM in order. If there is no residual space left for the chosen VM, then the heuristic for coldspot mitigation stops. Variance is defined as follows:

$$variance, R(t) = \frac{(mean - rescpu)^2 + (mean - resmem)^2 + (mean - resnet)^2 \ldots}{(m-1)} \quad (3)$$

$$mean = \frac{rescpu + resmem + resnet + \cdots}{m} \quad (4)$$

$$r_n = \sqrt{var_{p1}^2 + var_{p2}^2 \ldots + var_{pn}^2} \quad (5)$$

In above equation, mean is defined as the average of normalized residual capacities across 'm' different resources like cpu, memory, networks, etc. rescpu , resmem, resnet … stands for residual capacities across different resource dimensions. r_n is the magnitude of the vector which comprises of the individual variances across 'n' physical machines.

Khanna's Algorithm packs the VMs as tightly as possible trying to minimize the number of PMs by maximizing the variance across all the PMs. Thus, Khanna's algorithm minimizes power consumption by detecting underutilization in the managed using Max-Min thresholds selection model.

When the resource usage of a running PM violates a minimum predefined threshold value, the algorithm tries to pack the running VMs as close as possible thus trying to minimize the number of running physical machines.

C. Entropy:

Entropy proposes a consolidation algorithm based on constraint problem solving. The main idea of the constraint programming based resource manager is to formulate the VM resource allocation problem as constraint satisfaction problem, and then applies a constraint solver to solve the optimization problem. The ability of this solver to find the global optimum solution is the main motivation to take this approach. Entropy resource manager utilizes Choco constraint solver to achieve the objectives of minimizing the number of the running nodes and minimizing the migration cost. Entropy iteratively checks optimality constraint i.e. the current placement uses minimum number of the running nodes. If Entropy is successful in constructing a new optimal placement (uses fewer nodes) at VM packing problem (VMPP) phase, it will activate the re-allocation. Entropy employs a migration cost model that relates memory and CPU usage with migration context. High parallelism migration steps increases the cost. Using constraint programming techniques facilitates the task of capturing such context in two phases.

In the first phase, Entropy computes a tentative placement (mapping of VMs to PMs) based on the current topology and resource usage of PMs and VMs and reconfiguration plan needed to achieve the placement using minimum number of PMs required. In the second phase, it tries to improve the reconfiguration plan by reducing the number of migrations required. Since obtaining the placement and reconfiguration may take a considerable amount of time, the time given to the CSP solver is defined by the users, exceeding which whatever immediate value the solver has computed is considered for dynamic placement of VMs. VMs are classified as active or inactive based on their usage of CPU with respect to thresholds set.

The author define a viable configuration as one in which every active VM present in the cluster has access to sufficient cpu and memory resources on any PM. There can be any number of inactive VM on the PM satisfying the constraint. The CSP solver takes this viable condition into account in addition to the resource constraints, while procuring the final placement plan. However, considering only viable processing nodes and CPU-Memory Resource model is the limitation of the Entropy model [7] [25].

IV. EXPERIMENTAL TEST BED

Our implementation and evaluation is based on Xen Hypervisor. Four PMs with Xen 4.1 hypervisor installed were used to serve as the physical hosts and another machine was used as NFS [20] Server to house the VMs images. Physical machines which worked as clients were used simultaneously to generate load on the virtual machines hosted on the PMs. Seven VMs were created with Ubuntu 10.04, lucid host operating system with each 256 MB memory size and Ubuntu 11.10 as PMs Host operating system. They all have Apache, PHP and MySQL configured on them to act as web and Database servers. A separate machine has been configured which acts as the Management node, which runs the controller and the Decision Engine. The VIRT-M cluster management tool was implemented on this management node. Python programming was used to prototype these heuristics. Apart from these, RRD tool [22] was installed on the Management node running the Decision Engine, for storage of resource data. Our Experimentation takes these heuristics into consideration and implements these on idle VMs to investigate their impacts on performance of live migration in both source and target machine.

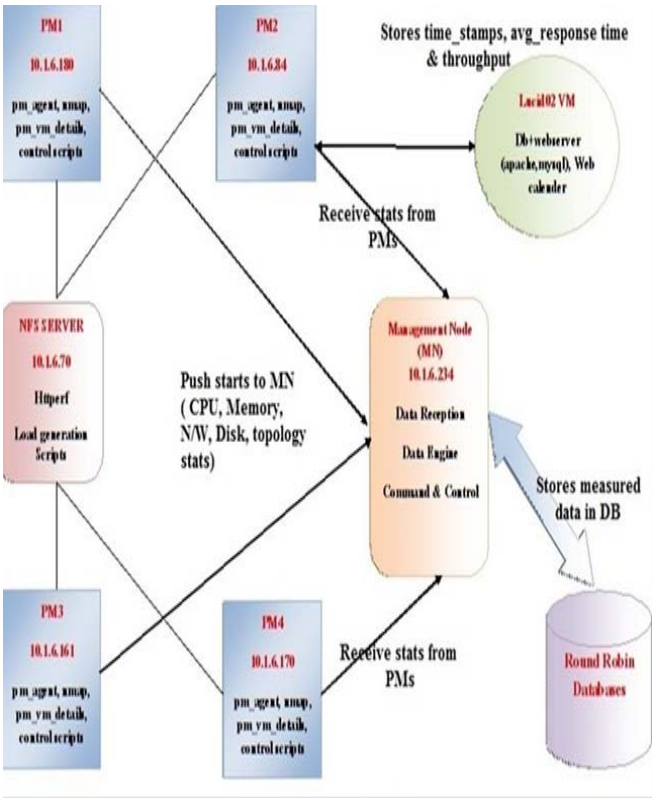

Figure 4.1 Experimental Test Bed

V. EVALUATION AND RESULTS

An experiment was performed with five VMs, two of which were idle and three having variable workloads. This is an interesting experiment to perform since if the rate is fixed, the cpu usage levels will not vary drastically within the duration of experiment and hence after some reconfiguration in the initial topology to start with, the PMs are expected to stabilize without further threshold violation. If the rate changes, the usage levels will vary more and in such a scenario, it will be of interest to have an idea how reactively the algorithms do consolidation. "Httperf" with Webcalender PHP scripts has been used for this experiment to inject varying workload. The following scenario was created:

PM84 - lucid08 - no load

PM161 - lucid12 and lucid13 - rates varying as 10 req/sec, 20 req/sec and 30 req/sec on both the VMs

PM170 - lucid09 - rates varying as 10 req/sec, 20 req/sec and 30 req/sec

PM180 - lucid14 - no load

The experiment was performed for 10 minutes duration for all the algorithms. The upper cpu threshold was kept as 60%. From figures 5.1, 5.2 and 5.3, it was observed that there is too much of cpu usage level variations which was expected due to variation of rates. Amongst the three, variation in Entropy has been slightly lesser amongst the PMs. Sandpiper mitigates 1 hotspot and triggers 1 migration from PM170 to PM180. Khanna's Algorithm performs 5 migrations in total and uses 3 PMs same as sandpiper.

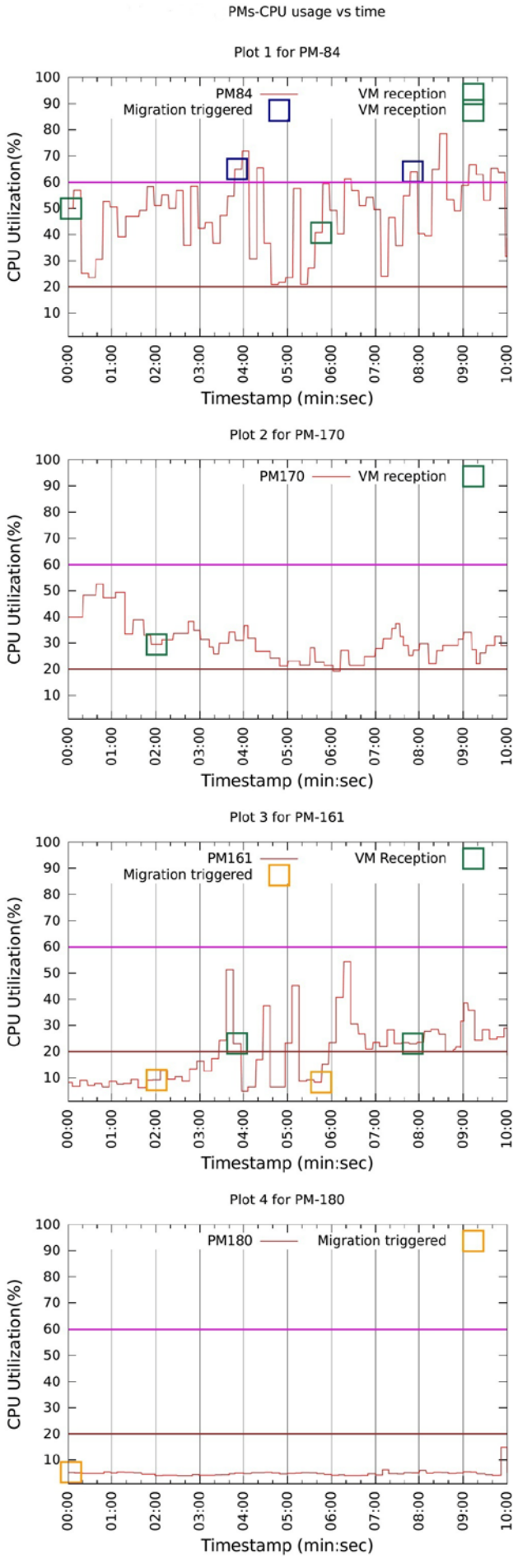

Figure 5.1 Khanna's Algorithm

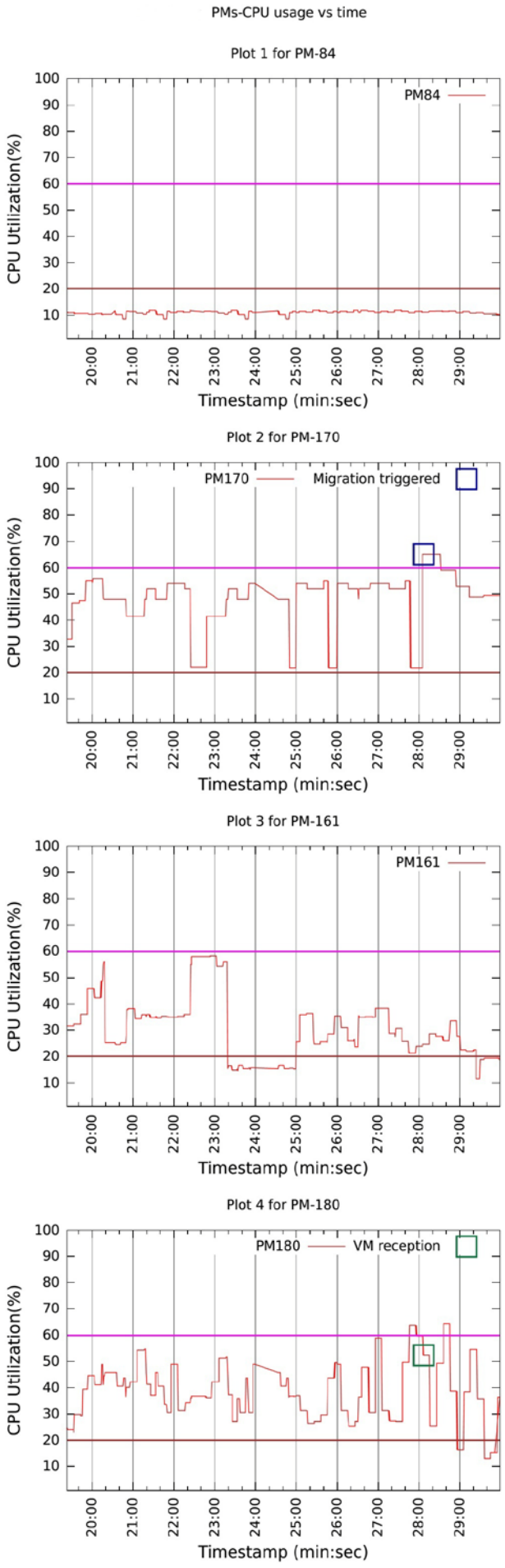

Figure 5.2 Sandpiper

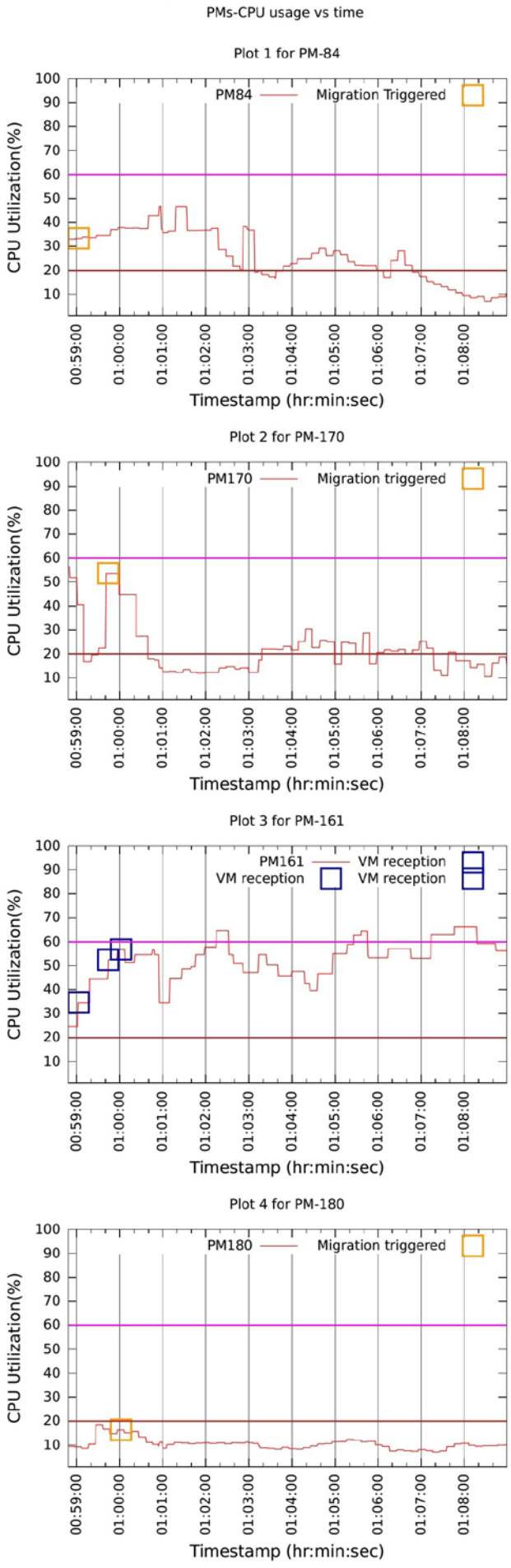

Figure 5.3 Entropy Algorithm

Entropy reduces the number of PMs by 3 and triggers 3 migrations. As it was analyzed that PM84 has comparatively higher usages than the others because of VM receptions from other PMs. From figures 5.1, 5.2 and 5.3, it was observed that Sandpiper detects a hotspot on PM170, migrates lucid09 to PM180 and reduces the number of PMs by 1. Khanna's Algorithm detects a coldspot at PM180, does coldspot mitigation. It triggers a series of migrations, and finally uses 3 PMs at the end of the experimental run. Lucid14 was migrated from PM 180 to PM 84. Lucid13 was migrated from PM161 to PM 170. Lucid 14 from PM84 to PM161, followed by lucid14 from PM161 to PM84, and finally lucid 08 from PM84 to PM161. Entropy in this case as well puts all the VMs on PM161. As already known in entropy, viable configuration is to be satisfied, which means number of active PMs should have access to sufficient available processing units. Here, even though the cpu usage of PM161 increases, the viability condition might have been satisfied for which all the VMs were put on the PM. Also, there was a decrease in cpu usage trend in the other PMs after entropy migrates all the VMs to PM 161.

The cpu usage of PM161 were expected to increase when varying workloads, but comparatively the hike is not that much. It was believed that this happened because when the rates are high on a VM, due to increase in processing time of the requests or delay in sending response to the client, the cpu usage of the VMs reduce even if the rate is high from the client. The new topologies generated are:

- Sandpiper - PM84 with lucid08, PM161 with lucid12 and lucid13, PM180 with lucid14 and lucid09
- Khanna's Algorithm - PM84 with lucid14, PM161 with lucid08 and lucid12, PM170 with lucid09 and lucid13
- Entropy - all on PM161

Table I describes the measured statistics. It was analyzed that both sandpiper and Khanna's Algorithm uses 3 PMs whereas Entropy uses just 1 PM. Sandpiper and Khanna's Algorithm reacts to hotspots when they are formed, whereas Entropy doesn't wait for hotspots or coldspots to happen, based on the current topology configuration, it searches for an optimal reconfiguration plan, once it finds it, the reconfiguration plan is implemented. That is why in this experiment entropy took only more than 1 minute to generate the new plan whereas, Sandpiper used 3 PMs after approximately 9.34 minutes from the start of the experiment and Khanna's Algorithm took more than 8 minutes to reduce the number of PMs. Also, Sandpiper triggers just 1 migration over Khanna's Algorithm and entropy.

Table I. Measured Evaluation Metrics

Algorithm	***No. of PMs***	***No. of Migrations***	***Time Taken(mins)***
Sandpiper	3	1	N/A
Khanna's Algorithm	3	5	8.22
Entropy	1	3	1.8

As mentioned earlier a plan has been designed to perform evaluation of application performance which is very crucial for analysis. But due to much unexpected circumstances valid correct response time values could not be logged from which some concrete results could be inferred. Here, the response time variation of 1 VM, lucid12 across all the three algorithms were presented.

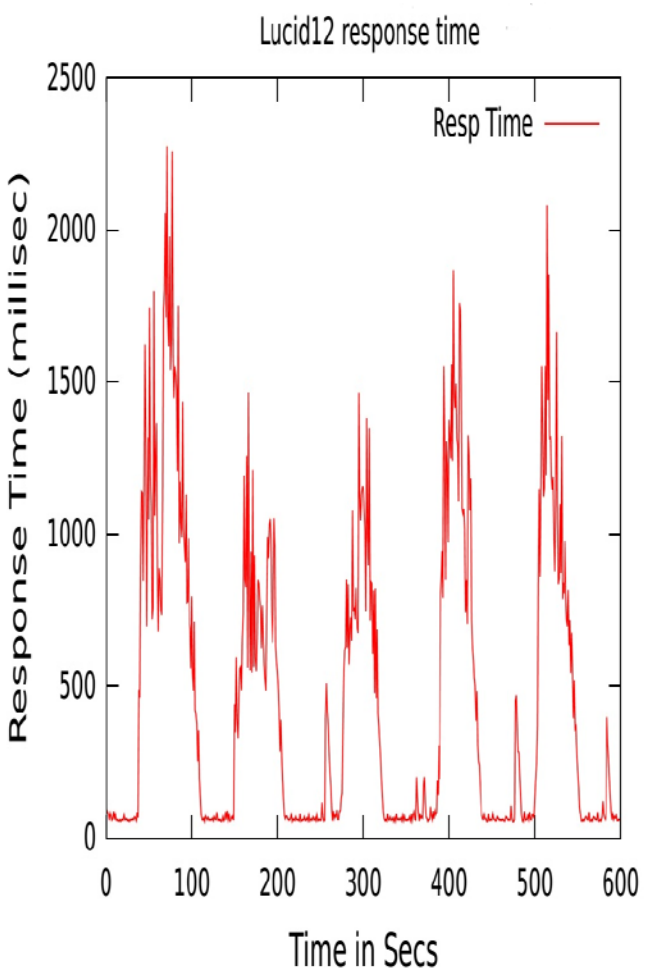

Figure 5.4 Response Time Variation of Lucid12 in case of Sandpiper

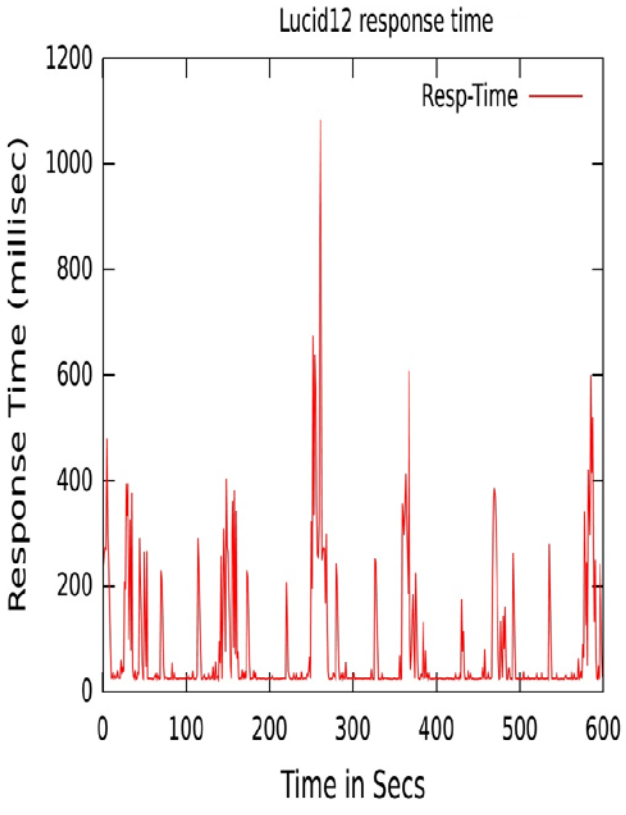

Figure 5.5 Response Time Variation of Lucid12 in case of Khanna's Algorithm

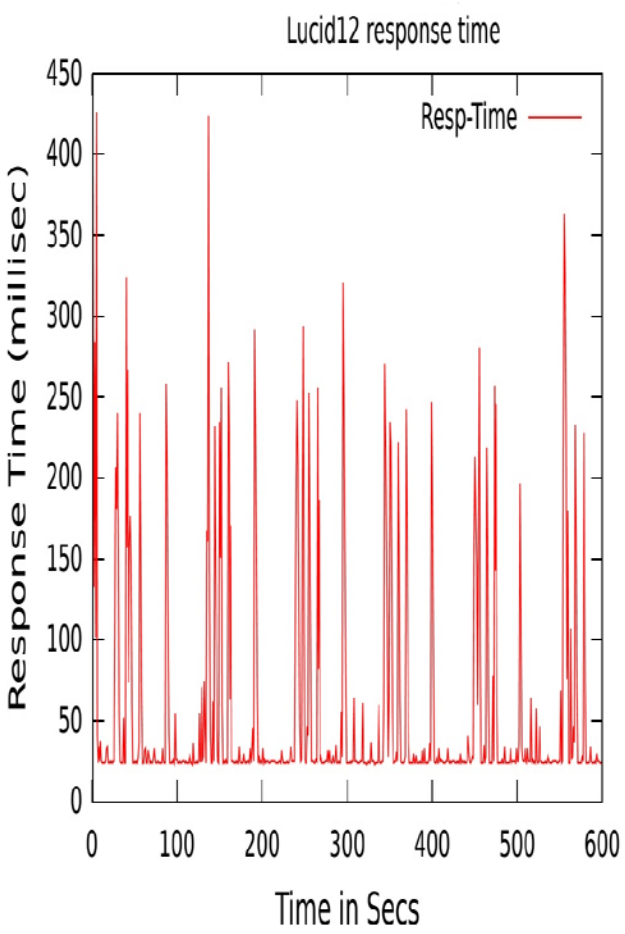

Figure 5.6 Response Time Variation of Lucid12 in case of Entropy

From figures 5.4, 5.5 and 5.6, it can be observed that response time for lucid12 in case of sandpiper is higher than Khanna's Algorithm and entropy. As seen from the graph, in experimenting with Khanna's Algorithm, the average response time was approximately 200 milliseconds with transient spikes where the response time value increased to as high as 600 millisecs and 1100 millisecs. The reason behind this sudden increase is the fact, that PM161 which hosted lucid12, triggered migrations of lucid13 (co-located VM). PM161 had been the source as well destination for three migrations. And it is known that the application performance of co-located VMs is affected in such cases. Reception and migration of VMs from the same host could occur where lucid12 resides must have been the reason of sudden spikes in the response time.

In case of Entropy, the results are in conformity to the events triggered by the algorithm. PM161 where lucid12 resides was chosen as the destination PM for all the VMs in the topology. As a result, PM161 had been the destination PM for three migrations. This increases the cpu usage of PM161 slightly affecting the response time at the server on lucid12. In addition to this, addition of three more VMs in lucid12's host and colocation amongst four other VMs must have affected its request processing, affecting its application performance. In Khanna's Algorithm there are infrequent spikes, but in entropy there are no spikes, the increase in response times are persistent, the average is very close to each other.

Unexpectedly, in sandpiper the response time values are more compared to both Khanna's Algorithm and entropy both of which are comparable. Although in Sandpiper,

lucid12 was not migrated neither did its host PM161 undergo VM receptions from other hosts, the response times are higher. This is because of the fact that, the MySQL server running at the VM lucid12 got blocked because of too many connections for some time. When the workload was generated with varied rates, since database was inaccessible for some time, the response time values increased and this had an effect on the overall response time values of the entire experimental run. However, it was noticed that the application performance of the VM gets affected if the host PM triggers too many migrations or undergoes VM receptions. A migrating VM will undergo degraded application performance. Thus, in this experiment, entropy does best in terms of application performance.

VI. CONCLUSION AND FUTURE WORK

With the popularity of cloud computing systems, live virtual machines migration will be great beneficial tool for dynamic resource management in the modern day data centers. To prevent server sprawl, server consolidation aims at reducing the number of server machines by consolidating load, enhancing resource utilization of physical systems along with provision of isolation & security of the application hosted. In this paper, we presented a performance evaluation of the chosen server consolidation algorithms in virtualized cloud computing environment when no workload is generated on the client machines. Sandpiper and Khanna's Algorithm uses a threshold based technique of triggering VM migrations. Entropy relies on CSP solver 4.4.1 to perform consolidation by providing a set of constraints, optimizing the number of PMs needed to house the VMs and the migration cost to determine the selection of configuration plan. In sandpiper, the migration cost is in terms of vsr metric whereas Khanna's algorithm considers the resource utilization as the migration cost metric. All of them intend to reduce migration cost in terms of the memory allocated to the VMs. Unlike other algorithms, Entropy tries to obtain a globally optimal solution, which distinguishes itself in its consolidation approach. Unlike other algorithms does, Entropy considers all the hosts in the topology and based on their current resource usages, finds out an optimal solution which tries to decrease the migration overhead in terms of memory. The other algorithms try to achieve consolidation on a per host basis, making sure that resource violations are prevented every time each host is scanned, and then the VMs are packed as closely as possible.

In case of varying workloads, when there are changing cpu utilization levels, entropy performs better than Khanna's Algorithm. It uses less number of PMs and triggers less number of migrations over Khanna's Algorithm. Moreover, it didn't take much time to procure the reconfiguration plan and initiate the new mapping. It is within acceptable limits to cross the upper cpu threshold, at the benefit of obtaining reduced number of PMs with less migration overhead. Too many migrations in the system always induces some amount of instability in the system, also, it could be observed that the relatively higher variation of cpu utilizations across the PMs with Khanna's algorithm than the others. The reasoning is that it is because of frequent VM relocation across PMs which happens in Khanna's Algorithm. Khanna's algorithm seems to be more befitting in an environment where there is need of cpu usages within specified thresholds as a high priority requirement, in the process of performing consolidation. It can be seen that better consolidation is possible than what Khanna's Algorithm does, but the cpu usage at the destination PM may reach too high, beyond acceptance level in some cases.

The application performance of lucid12 is best in Entropy. But unless it was analyzed the application performance of all the other VMs present in the topology, it cannot be ascertained concretely about the goodness of the algorithms. These algorithms try to efficiently prevent server sprawl and ensure non-disruptive load balancing in data centers. Efficiency of the algorithm depends on the resource parameters and metrics considered. Hence, a comparative performance analysis was carried out to analyse their applicability, goodness and incurred overhead. In near future, Evaluation of these algorithms with mixed load where different types of applications are used together, can facilitate in figuring out the distinct cases where an algorithm will behave well and hence can be used in those cases only to leverage the maximum benefits.

By increasing the scale of experimentation, investigations can be done in finding out whether behavior of algorithms changes when the number of PMs and VMs are increased in term of the metrics defined to compare the goodness of the algorithms. Moreover, more algorithms which does similar jobs like consolidation can be chosen in near future and their relative behavior can be analysed with the already chosen algorithms.

VII. REFERENCES

[1] A. Verma, P. Ahuja, A. Neogi, "pMapper: Power and migration cost aware application placement in virtualized systems", in Proceedings of the ACM/IFIP/USENIX 9th International Middleware Conference, 2008.

[2] Aameek Singh, Madhukar Korupolu, Dushmanta Mohapatra. "Server-Storage Virtualization: Integration and Load Balancing in Data Centers", in Proceedings of the SC 2008 ACM/IEEE conference on Supercomputing, 2008.

[3] Anju Mohan and Shine S, "Survey on Live VM Migration Techniques", International Journal of Advanced Research in Computer Engineering & Technology, vol 2, Jan. 2013.

[4] B. Speitkamp, M. Bichler, "A Mathematical Programming Approach for Server Consolidation Problems in Virtualized Data Centers", IEEE Transactions on Services Computing.

[5] C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfiled, "Live Migration of Virtual Machines", in Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation, vol. 2, pp. 286, 2005.

[6] Emmanual Arzuaga and David R. Kaeli, "Quantifying load imbalance on virtualized enterprise servers", in Proceedings of the first joint WOSP/SIPEW International Conference on Performance Engineering IEEE/ACM Trans. Netw. Pages 235-242, 2010.

[7] Fabein Hermenier, Xavier Lorca, Jean-Marc Menuad, Gilles Muller and Julia Lawall, "Entropy: A Consolidation Machine Manager for Clusters", in Proceedings of the 2009 ACM SIGPLAN/SIGOPS Internal Conference on Virtual Execution Networks, VEE, 2008, pp.41-50, 2009.

[8] G Jung, KR Joshi, MA Hiltunen, RD Schlichting RD, C Pu, "Generating adaptation policies for multi-tier applications in consolidated server environments", in Proceedings of the 5th IEEE International Conference on Autonomic Computing (ICAC 2008), Chicago, IL, USA, 2008; 23–32.

[9] G Jung, KR Joshi, MA Hiltunen, RD Schlichting RD, C Pu, "A cost-sensitive adaptation engine for server consolidation of multitier applications", in Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware (Middleware 2009), Urbana Champaign, IL, USA, 2009; 1–20.

[10] G. Keller and H. Lutfiyya, "Replication and migration as resource management mechanisms for virtualized environments", in Proceedings of the Sixth International Conference on Autonomic and Autonomous Systems (ICAS), pages 137-143, 7-13, 2010.

[11] Gunjan Khanna, Kirk Beaty, Gautam Kar and Andrzej Kochut, "Application Performance Management in Virtualized Server Environments", 10th IEEE/IFIP Conference on Network Operations and Management in Virtualized Server Environments, NOMS, 2006.

[12] James Greensky, Jason Sonnek, Robert Reutiman, and Abhishek Chandra, "Starling: Minimizing communication overhead in virtualized computing platforms using decentralized affinity-aware migration", 39th International Conference on Parallel Processing (ICPP), pages 228-237, September 2010.

[13] Jyothi Sekhar, Getzi Jeba and S. Durga, "A survey on Energy Efficient Server Consolidation through VM Live Migration", International Journal of Advances in Engineering and Technology, vol 5, pp.515-525, Nov. 2012.

[14] Kejiang Ye, Xiaohong Jiang, Dawei Huang, Jianhai Chen, and Bei Wang, "Live migration of Multiple Virtual Machines with Resource Reservation in Cloud Computing Environments" , in Proceedings of 2011 IEEE 4th International Conference On Cloud Computing, pp. 267-274, 2011.

[15] L. Kleinrock, "A Vision for the Internet", ST Journal of Research, 2(1):4-5, Nov, 2005.

[16] M. Bichler, T. Setzer, B. Speitkamp, "Capacity planning for virtualized servers", in Proceedings of the 16th Annual Workshop on Information Technologies and Systems (WITS'06), 2006.

[17] M Cardosa, M Korupolu, A Singh, "Shares and utilities based power consolidation in virtualized server environments", in Proceedings of the 11th IFIP/IEEE Integrated Network Management (IM 2009), Long Island, NY, USA, 2009.

[18] M. Nelson, B. Lim, and G. Hutchins, "Fast transparent migration for virtual machines", in Proceedings of the Annual Conference on USENIX Annual Technical Conference, pp. 25, 2005.

[19] N. Bobroff, A. Kochut, K. A. Beaty, "Dynamic placement of virtual machines for managing sla violations", in Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management (IM'07), 2007.

[20] NFS setup in Ubuntu, https://help.ubuntu.com/community/NFSv4Howto.

[21] NIST Definition of Cloud Computing v15, www.nist.gov/itl/cloud/upload/cloud-def-v15.pdf.

[22] RRD Tool, http://oss.octiker.ch/rrdtool/.

[23] S. Mehta, A. Neogi, "ReCon: A Tool to Recommend Dynamic Server Consolidation in Multi-Cluster Data Centers", in Proceedings of the IEEE Network Operations and Management Symposium (NOMS'08), Salvador, Bahia, 2008.

[24] S Srikantaiah, A Kansal, and F Zhao, "Energy aware consolidation for cloud computing", Cluster Computing 2009; 12:1–15.

[25] Susheel Thakur, Arvind Kalia and Jawahar Thakur, "Server Consolidation Algorithms for Cloud Computing Environment: A Review", International Journal of Advanced Research in Computer Science and Software Engineering, vol 3(9), September 2013.

[26] Timothy Wood, Prashant Shenoy, Arun Venkataramani and Mazin Yousif, "Sandpiper: Black-box and Gray-box Resource Management for Virtual Machines", Journal of Computer Networks, vol.53, pp.2923-2938, Dec.2009.

[27] V. Sarathy, P. Narayan, and Rao Mikkilineni, "Next Generation Cloud Computing Architecture- Enabling Real-time Dynamism for Shared Distributed Physical Infrastructure", 19th IEEE International Workshops on enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE, pp.48- 53, June 2010.

