Machine Learning based Optimized Live Virtual Machine Migration over WAN Links

Moiz Arif, Adnan K. Kiani, Junaid Qadir
School of Electrical Engineering & Computer Science (SEECS)
National University of Sciences & Technology (NUST), Islamabad, Pakistan
Email: {12mseemarif, adnan.khalid, junaid.qadir}@seecs.edu.pk

***Abstract*—Live virtual machine migration is one of the most promising features of data center virtualization technology. Numerous strategies have been proposed for live migration of virtual machines on Local Area Networks. These strategies work perfectly in their respective domains with negligible downtime. However, these techniques are not suitable to handle live migration over Wide Area Networks and results in significant downtime. In this paper we have proposed a machine learning-based downtime optimization (MLDO) approach which is an adaptive live migration approach based on predictive mechanisms that reduces downtime during live migration over wide area networks for standard workloads. The main contribution of our work is to employ machine learning methods to reduce downtime. Machine learning methods are also used to introduce automated learning into the predictive model and adaptive threshold levels. We compare our proposed approach with existing strategies in terms of downtime observed during the migration process and have observed improvements in downtime of up to 15%.**

Live Migration, Wide Area Network, Virtual Machine, Hypervisor.

I. Introduction

Virtualization is the art of abstraction of the hardware resources and provisioning them in software [1]. Multiple operating systems can run on single hardware resource shared amongst the instances. Hypervisors ensure that the resources are virtualized and provisioned to the virtual machines. Every virtual machine is given the illusion that it is in control of the entire hardware resource. Such features have enabled its mass deployment in data centers. Virtualization technology is being used to provide numerous services in every field of technology.

Maintenance of a data center involves servicing of servers, racks and network nodes. This requires the servers to be powered off for a pre-determined amount of time until the job is completed. In order to achieve this goal, the services hosted by server need to be moved to another location. This is known as migration of services. There are several important parameters linked with migration like total migration time, downtime of services and resource utilization. Migration of a virtual machine while it is running is known as live migration [2] [3]. The above mentioned performance parameters play a significant role in the live migration of services. Migrations can be planned as well as unplanned. The primary goal is to minimize downtime associated with live migration of virtual machines.

Local Area Network (LAN) is a network of computer systems that are limited to a small geographic area. Existing live migration techniques work well in a LAN environment [4]. Researchers have proposed many modifications and optimizations with the help of which live migration can be successfully completed without any noticeable service downtime. When it comes to Wide Area Networks, conditions change [4] and normal live migration techniques are not applicable anymore. The same techniques that work perfectly over LANs can cause significant downtime over WAN links causing the migration process to fail. WAN links are generally constrained in bandwidth and are often unpredictable. There is often limited bandwidth available with high latency, jitter and packet loss. These constraints impose certain restrictions that have to be catered for while designing migration strategies over WAN links.

In addition to the restrictions imposed by the WAN links, there are certain application level restrictions as well. VM accesses a memory for read or write operation and changes the content of the memory. The rate at which the VM changes the content of the memory is referred to as the page dirtying rate. It is an accepted norm that lower page dirtying rate would result in reducing the time required to iteratively copy the contents of memory. Size of the workload associated with VM also plays an important role since we generally do not have a common storage area network in WANs. The size of disk and memory can vary from a few hundred Megabytes to several Gigabytes and transferring such big files over limited-bandwidth links pose a serious challenge.

There are two commonly used migration strategies [4]: Pre-Copy [2] and Post-Copy [5]. In the pre-copy approach memory is copied from the source to destination in iterations until a certain threshold after which there is a brief stop in copy phase followed by the resume phase. In this strategy the downtime is time required for the stop and copy phase. Most hypervisors use the pre copy migration approach. In this approach we have to cater for the page dirtying rate. If the page dirtying rate is greater than the copy rate then this techniques is stuck into an infinite iterative loop. Initially the container or resource for VM is reserved at the destination. Subsequently, all memory pages are copied in the first round and successive rounds only copy the pages dirtied by the

VM. As a consequence, the source VM is suspended and certain measures are taken to fully synchronize the state of VM with destination. The state of VM at the source is then released and resumed at the destination.

The post copy strategy involves suspension of the VM at the start followed by the copy phase in which bare minimum processor state is copied at the destination. The processor state is sufficient to run VM. Subsequently, memory pages are copied to the destination. Here memory pages are not copied over and over again and each page is copied only once. Another noticeable difference is that in pre-copy approach the source handles all the requests; however, in post-copy this task is assigned to the destination. Different variations of post-copy technique have been proposed and they differ with each other in the way the memory pages are copied. The popular variations are Demand Paging, Active Pushing and Pre-Paging [6]. Demand Paging involves copying only the required page from the source. Active pushing copies the pages during suspension phase while the container is being set up at the destination. Pre-paging is an extension of active pushing as it predicts the requirement of a page by analyzing VM memory access patterns and based upon that, it changes the duration of the copy phase.

Another important factor in the migration process is the fact that the network connections or settings at the destination should be an exact replica of the source and should be consistent. In LAN networks this does not raise any significant concern as we remain in the same subnet and the IP addresses does not change. However, over WAN we can have different subnets and the downtime can be large enough to cause a TCP connection timeout and eventually breaks up the connection between server and the client. Various approaches like Provider Provisioned Virtual Private Network (PPVPN) [7], Active and Programmable Networks [8], Overlay networks [9] and Network Virtualization Environment (NVE) [10] have been proposed by researchers to tackle this problem.

Copying large workloads and VM disks over slow WAN links is an important concern. Apart from these challenges, we also have VMs with huge RAM allocations which pose a serious problem during live migration process. Researchers have proposed some solutions to this problem such as xNBD [11] and CR/TR-Motion [12]. Most approaches involve having a storage area networks where heavy workloads are copied independently without disrupting normal network traffic. Migrating traffic is assigned higher priority on the network. This can however result in overwhelming of link and causing service degradation to other applications.

Machine learning techniques are applied in computer sciences in order to reduce human intervention and make the system self-aware and automatic [27]. The learning process is based on data input over a period of time and the outcomes are used to make better decisions in the future. New data or samples are compared to the model optimized by the machine learning algorithms in order to categorize them and make decisions. For making efficient predictions, we expect the machine learning algorithms to produce prediction rules as accurately as possible. Machine learning can be applied to the domain of VM migration in data centers. In this particular scenario, the machine learning algorithms would be able to efficiently predict the time when the system should automatically migrate VMs away from any host in order to avoid significant downtime caused due to congestion of resources. These decisions are based on violation of threshold values set for physical and virtual parameters. Thresholds are referenced as the maximum values after which the system might show degraded performance. Threshold breach event can be avoided based on the predictions made by the machine learning algorithms. There are other advantages that can be harvested from such systems like conservation of energy by migrating VMs and shutting off extra servers. The main contribution of our work is to introduce automated learning into our predictive model using machine based learning techniques and decision tree learning.

Some salient aspects of our work:

A. Use of Prediction model:

In our work we make use of prediction methods in order to predict the conditions that result in triggering of the migration process. These prediction models identify the critical time intervals of resource utilization of the system where we have increased resource utilization. A migration during these critical time intervals can cause significant downtime. The prediction model takes into account the historical behavior of the system and makes sure that all the necessary migration of VMs are completed before the critical time intervals.

B. Use of Machine Learning Algorithms:

We introduce automated learning into our proposed model by employing machine learning algorithms. Machine learning introduces automated intelligence into the system with the help of which the system becomes self aware of the decisions to be made. With the help of prediction models these decisions are also scrutinized after any decision has been made. Our system is based on adaptive threshold based model for resources. The thresholds are continuously optimized overtime with he help of machine learning algorithms.

In this paper, we present a new migration strategy for WAN links. A new predictive algorithm MLDO is proposed that reduces downtime significantly. This paper is organized as follows: in section II we discuss the existing migration strategies. In Section III we present our proposed migration approach. Section IV describe results obtained from our migration approach. Section V sheds some light on future work and Section VI concludes the paper.

II. State of the Art Work

In [13], the authors have described a new approach to seamlessly migrate a live running virtual machine over Metropolitan Area Network/Wide Area Networks with no noticeable downtime. Xen virtualization environment is used with pre-copy migration approach. The downtimes quoted in the work were 0.8 to 1.6 seconds. These values are only 5-10 times larger than that of the LAN strategies. VM traffic controller is introduced which is basically a broker service and is responsible for provisioning of network, data, computational resources and IP tunnels. The strategy proposed in the paper does not take into account variable link speeds between nodes which can be a major bottleneck when evaluating service downtime.

In [14], authors have proposed a cooperative context aware approach towards data center migrations over WAN. They advertise the use to tunneling technology to immediately set up the network when the need for migration is identified. As soon as the migration process is completed, and having the network set up, the new traffic is now directed to the migrated VM via tunnel. The traffic is then re-routed to use a more direct and optimal path. For copying of the file systems, authors have proposed the use of flexible and adaptive replication system that replicates and synchronizes file system between local and remote locations. Asynchronous replication is used for the initial transfer of data in bulk and then switches to synchronous replication for the remainder of migration process. Authors have also discussed check pointing procedures for the migration scenario during unplanned outages. The paper lacks evaluations for variable link characteristics.

In [15], authors have proposed a complete platform for migration of virtual machine across WANs. A Virtual Private Network (VPN) based network infrastructure is used that cater for the networking requirements. Some optimizations have also been proposed for moving of memory and local persistent storage over bandwidth constrained WAN links. Extensive evaluations have been performed over software and in real world data centers and results have shown that total migration time is reduced by 30%, memory migration time is reduced by 65% and lowers the bandwidth consumption over the links for memory and storage migrations by 57%. Steps involved in live migration of VM include establishment of layer 2 connectivity between source and destination, transfer of disk, transfer of memory and suspend/resume phase. In order to achieve seamless migration of network connections, Virtual Private LAN Service (VPLS) based VPN technology is used in initial step. Asynchronous disk copy method is used and soon after synchronous copy mode for live memory migration is carried out. Some optimizations have been made to the pre-copy approach which improves performance and is termed as Smart Stop and Copy. Finally, content based redundancy method is used to conserve bandwidth.

In [16], authors have explained server consolidation frameworks and have given a detailed review and comparison of existing server consolidation frameworks and their challenges. Authors have described optimizations for virtual machine migrations and have compared the optimization techniques with each other.

In [17], authors have evaluated adaptive threshold based approach for the consolidation of virtual machines in cloud data centers. Authors have carefully explained how threshold based systems work and differences between different threshold based frameworks. Optimization to the adaptive threshold based algorithms gives the minimum possible Service Level Agreements (SLA) violation. The work done by the authors gives a detailed insight on threshold based systems and their contributions to the efficient consolidation of virtual machines in data centers.

In [18], authors have proposed an advanced live migration mechanism enabling instantaneous relocation of VMs based on post-copy live migration. Implementation has been done by including a light weight extension to KVM (Kernel-based Virtual Machine Monitor). Experimentation showed that a heavily-loaded VM was successfully migrated to another physical machine within 1 second.

After evaluating various techniques proposed for live migration of a virtual machine over WANs, we have concluded that these approaches are limited and are have a subset of features that a complete approach should have when we take wide area network links. A complete approach should be able to handle variable link speeds, unpredictable nature of traffic patterns, variable delays and in some cases disaster recovery procedures. The approaches lack predictive intelligence that can prevent disaster or proactively act in order to avert any unforeseen situation. Similarly, correlation between workload sizes, available bandwidth and nature of service offered by the application hosted by the VM is necessary for finding an optimal operating point before and during migration process.

III. Proposed Approach

In this paper, we propose predictive mechanisms to predict the need for migration by monitoring a distinct set of parameters. Our approach is divided into 2 modules: Monitoring Engine and Processing Engine. Both modules work in close coordination. The monitoring engine monitors physical and virtual parameters and collects data for the processing engine. The processing engine compares the data with pre-defined thresholds and apply machine learning methods to make decisions. We build a statistical model based on collected data and by varying parameters and link characteristics. Our approach is independent of underlying hypervisors and can run with any virtualization suite.

A. Monitoring Engine

The monitoring engine monitors CPU, memory and network utilization for servers and reports to the processing engine. This engine is also responsible to keep track of all usage of the server. The engine logs the data in the database for heuristic analysis. Based on the data being monitored, migration decision is taken. Various statistical data is logged and plotted for user-based decisions and automatic migrations. The idea here is to start the migration of VMs based on its usage patterns as well as the load conditions on the physical server. Figure 1 shows us the basic signaling phase for a threshold breach event. The monitoring engine on detecting a threshold breach event sends an exception to the processing engine which in turn gathers the required information necessary to continue with the migration process. Information is taken from the databases and the selected remote host is contacted in order to reserve the resources.

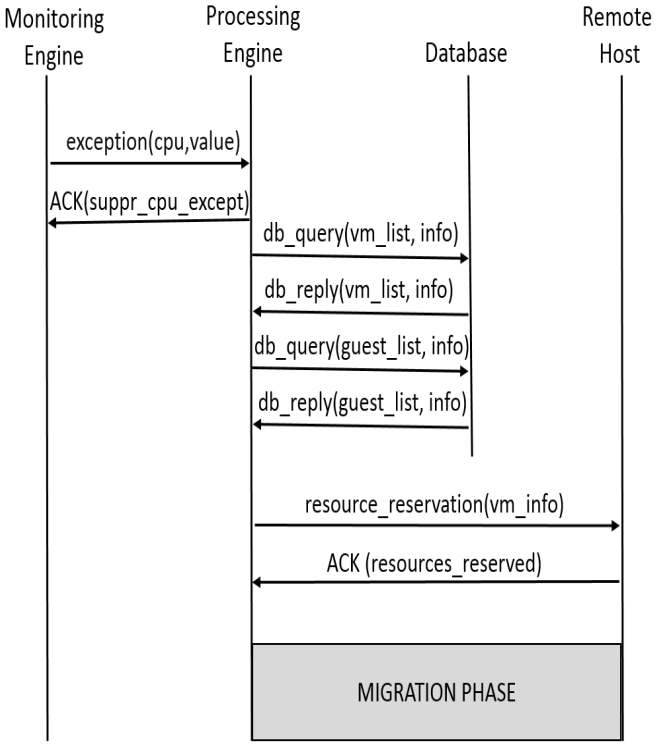

Fig. 1: A sample of signaling message flow of the monitoring engine due to threshold breach event

Parameters are monitored live and the data is saved in the database at 10 second intervals. The value of this interval can be changed keeping in view the required granularity and accuracy of monitoring. This data is saved later on by the processing engine in performing historical analysis and determining usage patterns for a particular node. Parameters monitored are CPU utilization, memory usage, network parameters and disk usage patterns. We leverage libvirt library and linux based modules for achieving the monitoring goals. For database operations we use MySQL. Monitoring engine also maintains a database of all available hosts in the data center along with all its available resources. It also stores information regarding all VMs running on hosts. This database is updated along with creation and deletion of VMs and with other related operations. The database holds the following information breached events, migration events, average utilization of hosts in the database along with timestamps, CPU, memory and network utilization, network parameters, disk usage patterns, list of all available hosts in the data center along with all its available resources, information on all VMs running on hosts. The database tables are updated row by row with new information being added to the respective tables using python MySQL APIs. When the engines start the Initial dimension is considered to be the current values of the entire system and as the time passes the averages historical information are included in the decision making process thereby fine tuning the system. The maximum allowed information is different for every table used in the MySQL database. The utilization values are capped at 7 days. Migration and breached events have no maximum allowed limitation. The VM and host tables are as per demand and based on the actual servers and VMs running in the datacenter.

The thresholds are defined and modified by the processing engine. Monitoring engine is responsible for correlating monitored data with the corresponding thresholds. In case of a threshold breach the monitoring engine raises a flag to the processing engine and specifics the parameter breached and other required information needed by the processing engine in making the migration decision. These decisions are made with the help of machine learning algorithm C4.5 [19]. The decision tree algorithm C4.5 is used to identify the need to migrate and similarly the prediction rules are defined. The decision tree is based on the CPU, memory and network threshold values. The cases where the thresholds have been breached are referred to as breached events. The cases where breached events are followed by migration is known as migration events. For both the cases respective flags are set and removed. Monitoring engine separately logs breached events, migration events and average utilization of hosts in the database along with timestamps. All threshold values are also stored in the database. Currently, historical data is being stored in the database for up to 7 days with varying frequency of updates depending on the parameter.

Threshold breach events are tackled in two possible scenarios. First scenario is Reactive handling of events in which a threshold breach event is detected by the monitoring engine and a flag is raised to the processing engine along with the necessary details about the threshold breach event. In this scenario the decision tree made by C4.5 tells us whether to migrate or not by comparing the current utilization values with the predefined thresholds and correlating with historical data available to the system. The second scenario is based on proactive handling of events which is further explained in the processing engine.We use C4.5 to generate the decision tree based on training data provided to the algorithm and

used for classification of the new arriving data set. Specific release is C4.5 Release 8 and use statistical pruning. From a statistical point of view the chances of getting false positives are reduced with time and as more and more historical data is available. The confusion matrix of the system shown in Table 2 is for the outcome of total 95 test cases. Here we define basic terms such as true negatives (15), true positives (73), false positives (4) and false negatives (3)

TABLE I: Confusion Matrix for the system

N = 95	Predicted 'No'	Predicted 'Yes'
Actual 'No'	15	4
Actual 'Yes'	3	73

The results from the confusion matrix shows that we have an Accuracy of 92.6% and a mis-classification rate of 7.4%.

The migration decisions specified at the end of the decision tree are not final. Final decision of migration is taken by the processing engine after analyzing the network and link characteristics at the source and the destination. Processing engine also performs heuristic analysis on the previous breach events and the average utilization values of the hosts during the estimated duration of the migration process.

Figure 2 shows us a sample decision tree for the case in which we take CPU and memory utilization parameters as a basis for decision making. The decision tree starts off with the CPU utilization on top due to the fact that without enough CPU resources available the migration process can fail and can cause performance degradation to other VMs or services running on the machine.

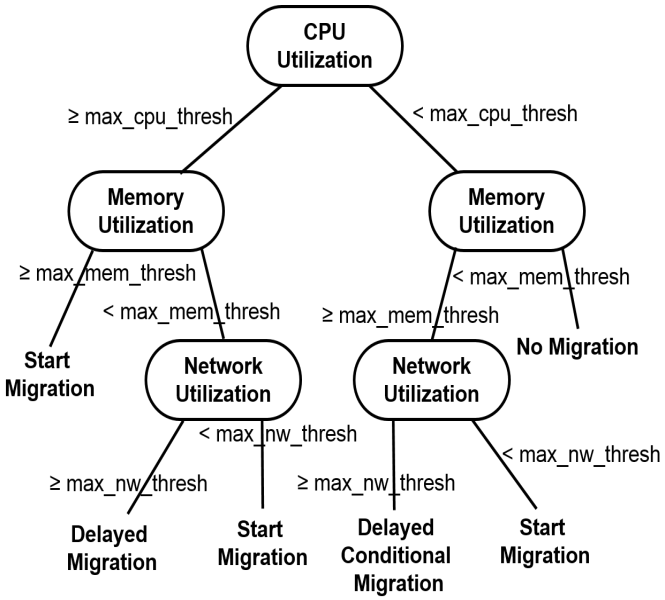

Fig. 2: Final decision tree diagram of the system which outputs the need to migrate a VM based on threshold breach events

The monitoring engine continues to monitor the parameters while the migration is in progress and also keeps a check on network utilization at both the source and the destination. Monitoring engines at both data centers talk to each other and share usage information on regular basis and also during the migration process. Monitoring engine acts as a bridge for external communication for any given host.

B. Processing Engine

The processing engine consumes usage data from the monitoring engine and performs migration of virtual machines based on thresholds. These thresholds are adaptive and change whenever a special conditions arises. Migration thresholds are set in order to avoid network congestion due to heavy migration traffic flow over WAN links. Whenever CPU load breaches a certain threshold the processing engine will check the network load and link condition. It will then decide on an adaptive migration traffic speed so that normal inter data center traffic is not affected. This technique is especially helpful when we have large RAM and disk sizes for VMs.

Thresholds for CPU utilization, memory utilization and network characteristics are defined in the processing engine and fed to the monitoring engine. These thresholds are adaptive and they change based on the usage patterns of the nodes and adjusting to an optimum value. Based on the number and time of occurrences of threshold breach events of any parameter and correlating it with usage patterns, the processing engine predicts with a certain degree of accuracy when on any given server there is a breach of threshold. This scenario is referred to as Proactive as compared to the Reactive scenario mentioned in the monitoring engine. With this prediction the processing engine takes preventive measures to avoid that situation. When the preventive measures are taken the processing engine does a post analysis of the situation and reinforces its decision or learn from it. Over time optimum values of thresholds is further polished. All of this is done by the C4.5 algorithm based on the training data provided to the processing engine by the monitoring engine and the database. The decision tree helps in the classification of new events. Current utilization values, historical average utilization, along with previous threshold breach and migration events along with timestamps are fed into the machine learning algorithm and it keeps on updating the thresholds dynamically. The initial threshold values are pre-defined in the processing engine source code and they are dynamically updated and optimized over the period of time.

The decision tree is a 2-tier structure which results in the migration decision. Delayed migration waits for adequate network bandwidth to be available before starting the migration process. Delayed conditional migration case is monitored more closely for some time until the memory utilization falls below the threshold or adequate network bandwidth becomes available. The no migration case describe

the case of continued migration of parameters under normal circumstances. Decision tree generation algorithm is explained in Algorithm 1.

Algorithm 1 Decision Tree Learning

```
if CPU − Util ≥ MaxCPUthresh then
    if Mem − Util ≥ MaxMemThresh then
        Start Migration
    else
        if Net − Util ≥ MaxNetThreshold then
            Delayed Migration
        else
            Start Migration
        end if
    end if
else
    if Mem − Util ≥ MaxMemThresh then
        if Net − Util ≥ MaxNetThreshold then
            Delayed Conditional Migration
        else
            Start Migration
        end if
    else
        No Migration
    end if
end if
```

Figure 3 shows the prediction process as a whole. It gives a high level understanding on how the monitoring and processing engines are inter linked and the types of data the processing engine consumes in order to make prediction. Custom data lookup refers to the fact that whenever a migration decision has to be made the processing engine queries the database for specific information that will help reinforce or cancel the hypothesis that a migration has to be made based on current threshold breaches. The historical data for a particular host/VM refer to the dataset saved in the database pertaining to a particular host/VM. The data is updated from time to time like a sliding window with the length of 7 days for utilization values. These values are co-related to the previous migration and threshold breach events to predict whether at any time period in the near future there would be a need to migrate workloads or not. For the initial case the reference time interval would be the time at which the VM boots up or the host comes online.

The prediction algorithm embedded in the processing engine takes the current utilization values (CUV), historical utilization averages (HUA) along with timestamps, previous threshold breach events (TBE) and previous migration events (PME) as input. The average utilization values are averaged over a 30 second interval. Further averages are calculated as per demand especially when comparing increased utilization valued intervals with the threshold breach identified cases. Threshold breach events and previous migration events

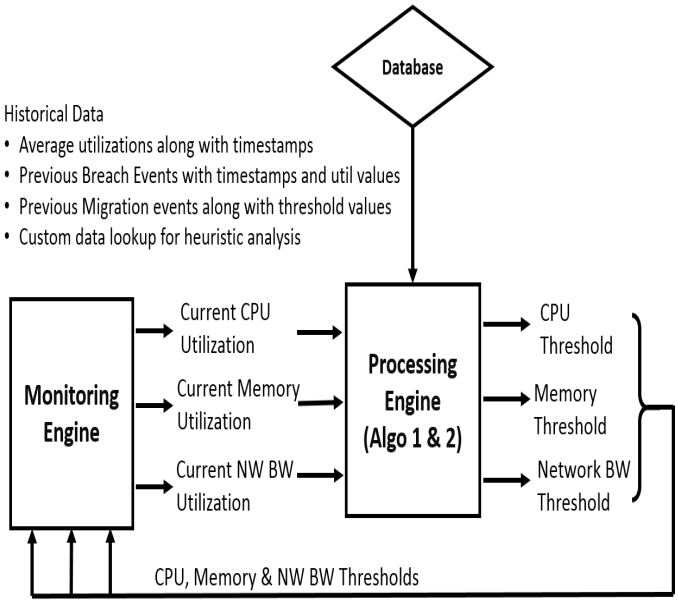

Fig. 3: System level block diagram showing calculation of adaptive thresholds and the overall prediction model used by the system

are mapped with each other to identify the cases where a threshold breach event resulted in migration of a VM. Such cases are referred to as threshold migration event (TME). Similarly, current utilization values and historical averages are mapped together and time intervals are identified where we have increased utilization averages. These intervals are referred to as increased utilization intervals (IUI). Then we identify critical time intervals (CTI) in which by mapping threshold migration events with these increased utilization intervals. Such intervals would be those where historically increased thresholds were recorded and the threshold breach event resulted in migration of VMs. The processing engine finds out the VMs to migrate and calculate the estimated time required to complete the migration event and executes the migration in such a way that it completes before the critical time interval approaches. Algorithm 1 will decide whether to migrate based on a breach event which happened due to a breached threshold however the results of this algorithm are used by the algorithm 2 to predict future migrations. Algorithm 2 learns from previous occurred events and adjust itself accordingly in order to make itself aware of the anomalies occurred in the past and tries to proactively avoid those conditions.

The internal working of the processing engine is based on the type of threshold breach notification it receives from the monitoring engine. This notification is taken as the need to migrate a virtual load away from a server. The first step is to choose a suitable destination amongst the list all available hosts. There is a scheduler routine running within the processing engine whose sole responsibility is to consume a list of hosts and output the most suitable host to run the virtual load. The decision tree includes taking into account

Algorithm 2 Prediction Algorithm

Require: current utilization values, historical utilization averages, threshold breach events, previous migration events

for all Values of CPU utilization, memory utilization and network utilization **do**
  $CUV \leftarrow current\_utilization\_values$
  $HUA \leftarrow historical\_utilization\_averages$
  $TBE \leftarrow threshold\_breach\_events$
  $PME \leftarrow previous\_migration\_events$
  while TBE & $PME \neq 0$ **do**
    for i *in* $range(len(TBE))$ **do**
      for j *in* $range(len(PME))$ **do**
        if $TBE[i][j] == PME[i][j]$ **then**
          $TME \leftarrow a[i]$
        else
          continue
        end if
      end for
    end for
  end while
  for k *in* $range(len(CUV))$ **do**
    for l *in* $range(len(HUA))$ **do**
      if $CUV[k][l] > HUA[k][l]$ **then**
        $IUI \leftarrow HUA[k][l]$
      else
        continue
      end if
    end for
  end for
  for m *in* $range(len(TME))$ **do**
    for n *in* $range(len(IUI))$ **do**
      if $CUV[k][l] == HUA[k][l]$ **then**
        $CTI \leftarrow IUI[k][l]$
      else
        continue
      end if
    end for
  end for
end for

the resource table maintained by the monitoring engine in the database. The scheduler makes sure to avoid migrating a VM to a heavily-loaded server or to a server where for instance the CPU utilization is bound to increase based on the usage patterns of that server. Simultaneously, choosing which virtual machines to migrate is based on the minimization of migration [17] approach. This approach ensures that minimum number of VMs are migrated from the source to the destination. After each migration is done the threshold flags are checked, the migration process continues until the threshold breach event has been closed.

As soon as appropriate destination is identified by the scheduler, the required resources are reserved at the destination and the database is updated accordingly. Now the migration process starts and it uses Pre-Copy technique. As a first step the disk contents are copied at the destination and then the memory contents. Memory is copied in iterations and when the remaining dirty pages is below a specified threshold the VM is shut off to copy the remaining contents and then VM is resumed at the destination. The traffic is temporarily routed from the host to the destination via tunnels in order to keep the incoming network connections alive till the time the new route is advertised for the new destination. Processing engine maintains a routing table of all available routes for all virtual machines running on physical machines along with other important information.

Another important factor is the network bandwidth utilized by the migration process. Migration traffic is high priority and it takes preference over normal traffic. Processing engine monitors the traffic patterns on the link and controls the bandwidth allocated to the migration process so that the migration process doesnt overwhelm the link and cause service disruption for other services. This consideration applies at the source and at the destination. Figure 4 shows the initialization phase for the proposed system.

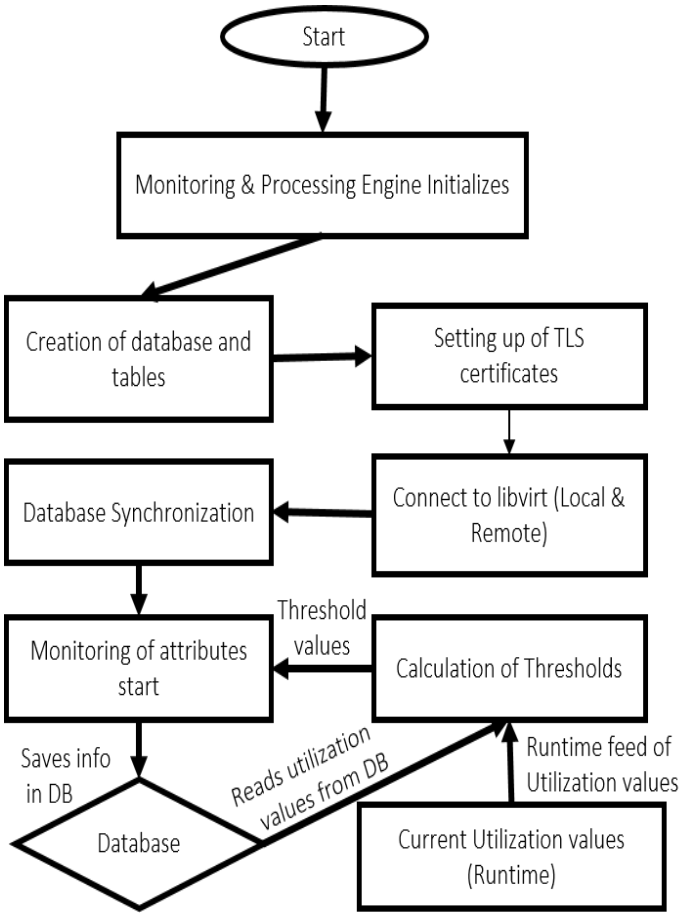

Fig. 4: Initialization phase for monitoring and processing engines

IV. Results

We performed extensive experimentation using different workloads to test our proposed approach. Before migration we ran server application on the VM that listens to client requests on TCP sockets. We ran client application on a standalone machine which is sending messages to the Server VM and getting response in return. Client and server connection was setup and messages were exchanged. Our algorithm starts migration when the resource utilization exceeds the threshold. In this case we observed that migration was triggered by processing engine after receiving a threshold breach alarm from monitoring engine. Migration caused an increased in CPU utilization by 35% and memory utilization by 10%. Migration traffic was sent at a constant rate and in order to decrease downtime during the stop and copy phase the network bandwidth was momentarily increased by 3 times. It was observed that TCP connection did not break and the packet stream was received in order with negligible downtime.

A. Testbed Setup

We have performed migrations over local systems having 2.5 GHz core i5 processors and 4 GB of RAM. We have used netem [20] tool along with ethtool to simulate WAN environment for migrations. Experiments have been performed for lightly-loaded and heavily-loaded servers. We have a client/server application running on the VM that receives requests from clients and sends a response. We plan to measure the time duration in which the clients do not receive a response from the server. VM has a total disk size of 10 Gigabytes pre-allocated with a 1 Gigabyte RAM and a full Ubuntu 12.04 LTS desktop is running on top of it. Network link between the two machines is 1G which is later changed for experimentation. For experimentation, we have varied the CPU utilization of the workload in order to verify the working of our algorithm. We have also experimented with different types of workloads and observed the downtime in each case. We have also performed experiments on different types of workloads including web server and File Transfer Protocol (FTP) servers. For the comparison section we have replicated various scenarios to the best of our abilities and changed the configurations of the above mentioned test-bed system.

B. Comparison with existing strategies

We have tested our framework on 2, 3 and 4 servers in order to test how the system scales when more devices are added to the pool of available hosts. The framework is self-sustaining and it can cater for as many number of servers in the data center. The results were consistent for all the three test cases. In [18] authors report total migration time of 65 seconds when using Pre-Copy and 10 seconds when using optimized Post-Copy migration approach. These experiments were performed with dual 1G links and with VM running at 100% capacity. However, when we apply constraints on the links in order to simulate WAN links the above results degrade significantly. With the same environment and VM specifications our system completes the migration process using pre-copy technique in 58 seconds. It is worth mentioning that when WAN characteristics were applied to the network link the migration time increased to 4 minutes which indicates that WAN links have an unpredictable and a significant impact on application performance.

The survey of adaptive threshold techniques and optimizations mentioned in [16] and dynamic consolidation of VMs based on adaptive utilization thresholds [17] techniques give us an insight to how the threshold based systems work. The techniques explained by the authors cater for one parameter at a time. In our system we incorporated CPU, memory and network based thresholds in order to avoid a failed migration. As explained earlier a fully utilized network link can cause delays for the migration traffic as well as other application running on the server as well as for the workloads themselves. Figure 5 shows the results of the above comparisons.

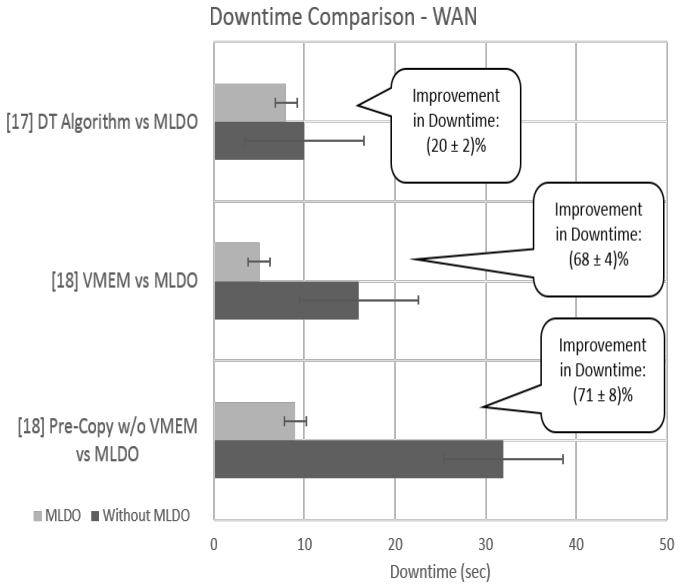

Fig. 5: A comparison of downtimes offered by DT Algorithm [17], VMEM [18] and MLDO

The optimized downtime shown in the Figure 5 are averaged values. The optimized downtime [18] pre-copy technique is 9 ± 0.6 seconds with 95% confidence interval, [18] post-copy 5 ± 0.1 seconds with 95% confidence interval and [17] is 8 ± 0.5 seconds with 95% confidence interval. The confidence interval calculations are done following equations from [22]. The range of downtime signifies the variable and unpredictable nature of the link. Contributing factors are bursty traffic, variable end to end latency and other congestion related phenomenon. In each case our proposed

systems performs better and offers lesser downtime.

The downtime values mentioned by the authors above vary significantly when we apply WAN link characteristics on the links connecting the host to the destination. Figure 6 shows the effects of increasing latency on the downtime.

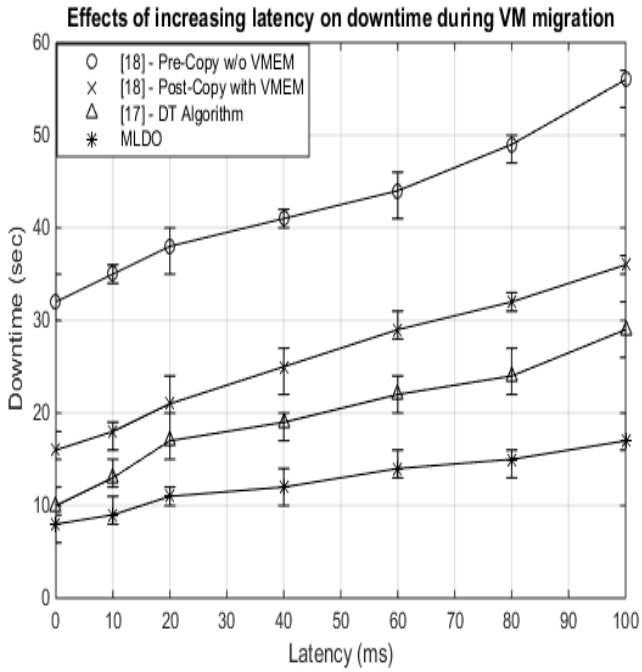

Fig. 6: Effects of increased latency on downtime offered by DT Algorithm [17], VMEM [18] and MLDO

C. Effects of variable Latency on the system

We performed experimentation using a VM having a 10GB disk size and 1GB RAM size. It was observed that by increasing the latency there was an increase in downtime and total migration time. Similarly, a greater change in both the metrics was observed in the case for variable latency. Web server application was running in this experiment. The migration and service downtime were measured using original migration strategies of KVM [21]. Processing engine caters for the variable latency on the link and adjusts network speed of the migration traffic accordingly and ensures that migrations process does not affect other services. During the final stop and copy of the RAM dirty pages processing engine ensures maximum transfer rates. Migration time increases with the increase in latency on the link. Figure 7 shows the effect of latency on the migration process.

D. Effects of variable Threshold levels on the system

During experimentation we noticed that migrations causes as increase in the CPU and memory utilization on the physical machine. The results are given in Table 2 which represent that by varying the CPU and memory thresholds we achieve

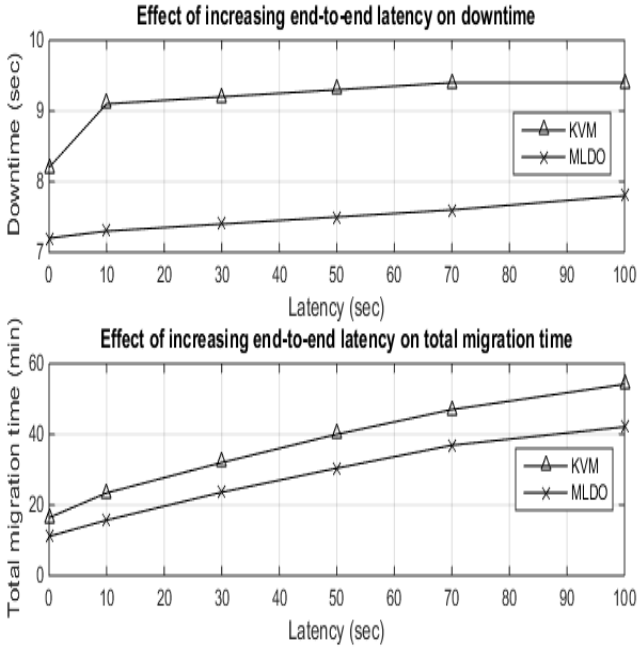

Fig. 7: Effects of increased end-to-end latency on downtime offered by KVM [21] and our approach

different results. When the threshold levels are increased the CPU get to a phase where it gets fully loaded, this means that processes will have to wait a longer period of time in order to get hold of the CPU which increases the downtime. In Figure 8 we see the effects of these thresholds on the downtime.

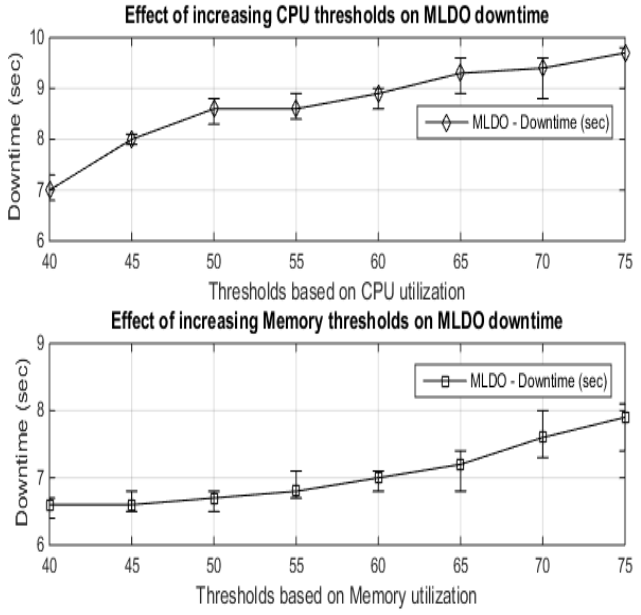

Fig. 8: Effects of varying CPU and memory utilization thresholds on downtime offered by MLDO

Table 2 shows the effects of CPU and memory thresholds on the system and the improvements different threshold levels give us.

TABLE II: Threshold Information and Improvements

CPU Threshold limit	% Increase in CPU utilization during migration	Memory Threshold limit	% Increase in Memory utilization during migration	% Improvement in downtime
50%	$\sim 35\%$	60%	$\sim 10\%$	$10 \pm 2\%$
55%	$\sim 35\%$	65%	$\sim 10\%$	$7 \pm 2\%$
60%	$\sim 35\%$	70%	$\sim 10\%$	$5 \pm 2\%$
65%	$\sim 35\%$	75%	$\sim 10\%$	$3 \pm 1\%$

We compared our migration algorithm with standard migration algorithms of KVM [21]. These standard algorithms use pre-copy approach and are built into the hypervisors. We apply our proposed algorithm on top of these existing techniques and compare results. We logged migration time and service downtime by changing network parameters of variable latency and variable link speeds. We then carried out a series of migration experiments by changing the parameters of link speeds and latency. Our algorithm ensured minimum downtime for all such scenarios. We calculated downtime by timing ICMP echo request and reply messages. Figure 7 give the results of performing the same series of experiments with our proposed framework. Standard experimentation includes migration with off the shelf techniques which are coded in the hypervisor and virtualization libraries by default. After implementing our proposed algorithm we managed to reduce downtime and total migration time. We also noticed that overall migration time is reduced for low latency links and remains fairly constant for high latency links.

E. Effects of variable link speeds on the system

We adjusted the transfer speeds to reduce downtime as well as total migration time. Our prediction algorithm initiates migration at an optimum point by monitoring all the parameters. Improvements can be observed in downtime as well as overall migration time. This is due to the variable adaptive rate of transmission used during migration. Migration traffic has a higher priority as compared to normal network traffic. Our proposed approach increases the transfer speed to an extent where the normal traffic does not get affected. Whenever network traffic is increased, the transmission rates adjusts accordingly. Subsequently, we found an optimized migration point by conducting a series of experiments by changing parameters of CPU utilization, memory utilization and link characteristics. We also vary link speed, latency and packet loss. Figure 9 shows the effect of increasing link speed on the downtime and total migration time.

F. Effects of data compression on the system

Data compression methods are used in order to reduce the size of data. In our scenario data compression reduces

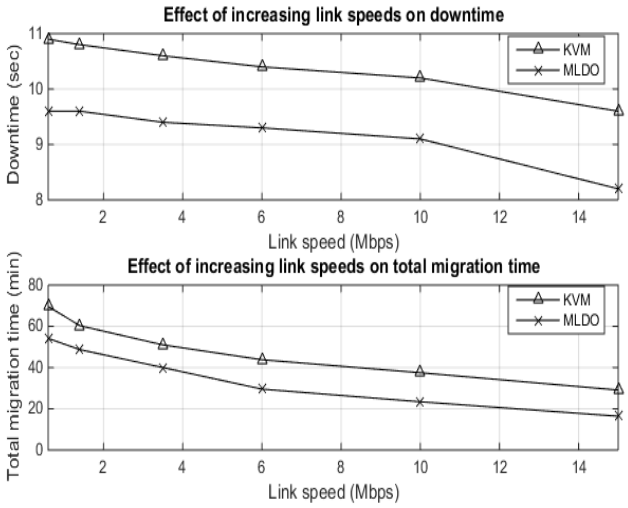

Fig. 9: Effects of increased link speed on downtime and total migration time

the number of bytes sent from the source to the destination during the migration process. There are many compression algorithms available such as Lempel-Ziv 77 (LZ77) [23], Lempel-Ziv-Welch [24], Huffman codes [25] and etc. These compression algorithms can offer compression of upto 40% or more [26]. We used LZ77 algorithm to observe the effect of data compression on live migration. In our case we have a relatively large data to send over the link. Although data compression algorithms save bandwidth but they also increase the overall migration time due to extra processing involved at the source and destination for compression and decompression respectively. After experimentation we observed that the overall size of data was reduced by up to 35% due to compression. The effect on downtime was negligible, however total migration time varied depending on the size of the workload. Figure 10 shows the effect of compression using LZ77 algorithm on MLDO.

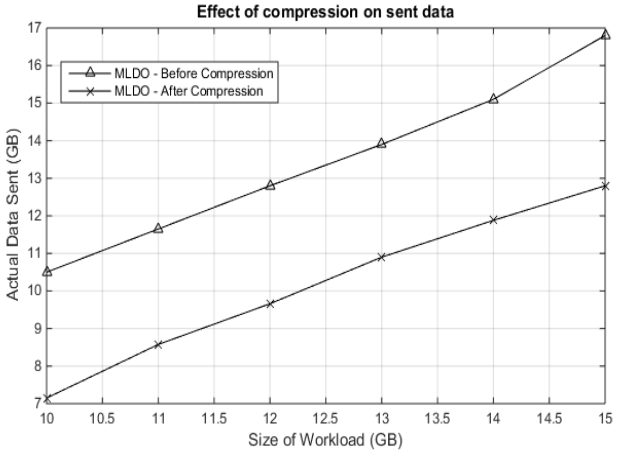

Fig. 10: Effects of compression using LZ77 algorithm on data bytes sent using MLDO

V. Future Work

In the future we plan to develop a signaling framework which will cater for inter-hypervisor migrations. This is important when there are more than one type of hypervisor being used and to cater for this we must have a system that can communicate and convey information between different hypervisors. We plan to conduct extensive testing and implement signaling framework to go along with processing and monitoring engines and with more workload types. Our current model reduces the overall downtime of VM services during live migration. We believe that a standalone all in one package is the solution for problems related to live VM migration across WAN. A single entity should be responsible for co-relating various parameters and taking optimal decisions which would reduce downtime. Our platform can be placed alongside SDN controllers at a centralized location and can also be ported out to use with OpenFlow. We plan to conduct extensive testing and optimization on bandwidth conservation in order to conserve more bandwidth and reduce downtime and total migration time.

VI. Conclusion

Live virtual migration is an important tool having many advantages in both LAN and WAN environments. Live migration involving WANs pose some new challenges that have to be accounted for in order to achieve seamless migration of a VM without significant downtime. Moving large workloads with high page dirtying rate while keeping the network configurations consistent can be very complicated. In addition, factors such as memory size, page dirtying rate, network transmission rate and migration algorithms directly affect the performance of live migration. In this paper we have reviewed some of the migration techniques. We proposed an adaptive live migration approach based on prediction and machine learning algorithms called MLDO. MLDO reduces total migration time and downtime by making the system intelligent and adaptive. Compression algorithm was employed in order to reduce the amount of data sent during the migration process. We have quoted an improvement of about 15%. MLDO is adaptive in nature and it caters for variable link characteristic on WAN links.

References

[1] Barham, Paul, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. "Xen and the art of virtualization." *ACM SIGOPS Operating Systems Review 37,* no. 5 (2003): 164-177.

[2] Clark, Christopher, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. "Live migration of virtual machines." *In Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2,* pp. 273-286. USENIX Association, 2005.

[3] Medina, V., & Garca, J. M. (2014). "A survey of migration mechanisms of virtual machines." *ACM Computing Surveys (CSUR), 46(3),* 30.

[4] Kapil, D., Pilli, E. S., & Joshi, R. C. (2013, February). "Live virtual machine migration techniques: Survey and research challenges." *In Advance Computing Conference (IACC), 2013 IEEE 3rd International,* (pp. 963-969). IEEE.

[5] Hines, M. R., Deshpande, U., & Gopalan, K. (2009). "Post-copy live migration of virtual machines." *ACM SIGOPS operating systems review, 43(3),* 14-26.

[6] Hines, M. R., & Gopalan, K. (2009, March). "Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning." In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments* (pp. 51-60). ACM. Hines, Michael R., Umesh Deshpande, and Kartik Gopalan. "Post-copy live migration of virtual machines." ACM SIGOPS operating systems review 43.3 (2009): 14-26.

[7] Andersson, L., & Madsen, T. (2005). *Provider Provisioned Virtual Private Network (VPN) Terminology* (No. RFC 4026).

[8] Campbell, A. T., De Meer, H. G., Kounavis, M. E., Miki, K., Vicente, J. B., & Villela, D. (1999). "A survey of programmable networks." *ACM SIGCOMM Computer Communication Review, 29(2),* 7-23.

[9] Andersen, D., Balakrishnan, H., Kaashoek, F., & Morris, R. (2001). *Resilient overlay networks* (Vol. 35, No. 5, pp. 131-145). ACM.

[10] Chowdhury, N. M. K., & Boutaba, R. (2010). "A survey of network virtualization." *Computer Networks, 54(5),* 862-876.

[11] Hirofuchi, T., Ogawa, H., Nakada, H., Itoh, S., & Sekiguchi, S. (2009, May). "A live storage migration mechanism over wan for relocatable virtual machine services on clouds." In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid* (pp. 460-465). IEEE Computer Society.

[12] Liu, H., Jin, H., Liao, X., Yu, C., & Xu, C. Z. (2011). "Live virtual machine migration via asynchronous replication and state synchronization." *parallel and distributed Systems, IEEE Transactions on, 22(12),* 1986-1999.

[13] Travostino, F., Daspit, P., Gommans, L., Jog, C., De Laat, C., Mambretti, J., Monga, I., Van Oudenaarde, B., Raghunath, S., & Wang, P. Y. (2006). "Seamless live migration of virtual machines over the MAN/WAN." *Future Generation Computer Systems, 22(8),* 901-907.

[14] Ramakrishnan, K. K., Shenoy, P., & Van der Merwe, J. (2007, August). "Live data center migration across WANs: a robust cooperative context aware approach." In *Proceedings of the 2007 SIGCOMM workshop on Internet network management* (pp. 262-267). ACM.

[15] Wood, T., Ramakrishnan, K., van der Merwe, J., & Shenoy, P. (2010). "Cloudnet: A platform for optimized wan migration of virtual machines." *University of Massachusetts Technical Report TR-2010-002* .

[16] Ahmad, R. W., Gani, A., Hamid, S. H. A., Shiraz, M., Yousafzai, A., & Xia, F. (2015)."A survey on virtual machine migration and server consolidation frameworks for cloud data centers." *Journal of Network and Computer Applications, 52,* 11-25.

[17] Beloglazov, A., & Buyya, R. (2010, November)."Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers." In *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science* (p. 4). ACM.

[18] Hirofuchi, T., Nakada, H., Itoh, S., & Sekiguchi, S. (2010, May). "Enabling instantaneous relocation of virtual machines with a lightweight vmm extension." In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on* (pp. 73-83). IEEE.

[19] Quinlan, J. R. (1993). C4. 5: Programs for Machine Learning.

[20] Hemminger, S. (2005, April). Network emulation with NetEm. In *Linux conf au* (pp. 18-23).

[21] http://www.linux-kvm.org/page/Migration

[22] Box, G. E., Hunter, W. G., & Hunter, J. S. (1978). Statistics for experimenters

[23] Ziv, J., & Lempel, A. (1977). "A universal algorithm for sequential data compression." *IEEE Transactions on information theory,* 23(3), 337-343.

[24] Kida, T., Takeda, M., Shinohara, A., Miyazaki, M., & Arikawa, S. (1998, March). "Multiple pattern matching in LZW compressed text. In " *Data Compression Conference, 1998. DCC'98. Proceedings* (pp. 103-112). IEEE.

[25] Sharma, M. (2010). "Compression using Huffman coding. " *IJCSNS International Journal of Computer Science and Network Security, 10(5),* 133-141.

[26] Cui, Lei, Jianxin Li, Bo Li, Jinpeng Huai, Chunming Hu, Tianyu Wo, Hussain Al-Aqrabi, and Lu Liu. "VMScatter: migrate virtual machines to many hosts." In *ACM SIGPLAN Notices,* vol. 48, no. 7, pp. 63-72. ACM, 2013.

[27] Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (Eds.). (2013). "Machine learning: An artificial intelligence approach." *Springer Science & Business Media.*