Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 54 (2015) 167 – 176

Procedia Computer Science

Eleventh International Multi-Conference on Information Processing-2015 (IMCIP-2015)

Virtual Machine Migration Triggering using Application Workload Prediction

Bane Raman Raghunath* and B. Annappa

Department of Computer Science and Engineering, National Institute of Technology Karnataka, Surathkal 525 075, India.

Abstract

Dynamic provisioning of physical resources to Virtual Machines (VMs) in virtualized environments can be achieved by (i) vertical scaling-adding/removing attached resources from existing virtual machine and (ii) horizontal scaling-adding a new virtual machine with additional resources. The live migration of virtual machines across different Physical Machines (PMs) is a vertical scaling technique which facilitates resource hot-spot mitigation, server consolidation, load balancing and system level maintenance. It takes significant amount of resources to iteratively copy memory pages. Hence during the migration there may be too much overload which can affect the performance of applications running on the VMs on the physical server. It is better to predict the future workload of applications running on physical server for early detection of overloads and trigger the migration at an appropriate point where sufficient number of resources are available for all the applications so that there will not be performance degradation. This paper presents an intelligent decision maker to trigger the migration by predicting the future workload and combining it with predicted performance parameters of migration process. Experimental results shows that migration is triggered at an appropriate point such that there are sufficient amount of resources available (15–20% more resources than high valued threshold method) and no application performance degradation exists as compared to properly chosen threshold method for triggering the migration. Prediction with support vector regression has got decent accuracy with MSE of 0.026. Also this system helps to improve resource utilization as compared to safer threshold value for triggering migration by removing unnecessary migrations.

Keywords: Live virtual machine migration; Resource provisioning; Workload prediction; Support vector regression.

1. Introduction

Cloud computing is attracting the organizations to use virtualized resources in the cloud data centers as it enables multiple applications to collocate with total isolation on a single physical machine. It also enables dynamic resource provisioning through virtual machine resizing and live virtual machine migration to eliminate hotspot, consolidating servers on fewer virtual machines and load balancing. These provisioning techniques are of two types i.e. vertical scaling and horizontal scaling[1]. Vertical scaling allocates or de-allocates resources to or from existing virtual machine and horizontal scaling adds or removes virtual machine instance. Virtual machine migration comes under vertical scaling as resources are de-allocated from one and added to the newly created virtual machine at another physical machine. Xen[2] uses pre-copy mechanism as virtual machine migration technique in which memory pages are

*Corresponding author. Tel.: +91-895-182-4006.
E-mail address: ramanbane.cs12f02@nitk.edu.in

transferred iteratively due to page dirtying process. Pre-copy combines push and stop-and-copy phases. In push phase the source VM continues running while certain pages are transferred to the destination. To ensure consistency, pages modified during this process are resend. In stop-and-copy phase the source VM is stopped, remaining dirtied pages are copied to the destination VM and then new VM is started at destination.

HotSpot handling and Server Consolidation are the two general scenarios that appears in data centers. HotSpot are detected when a virtual machines load level increases. At this time it requires additional amount of resources and additional allocation should be done to satisfy the required performance level. Server Consolidation is done when a virtual machines load level decreases and results in under-utilization of allocated resources. Hence, the extra resources can be removed and Virtual Machines (VMs) can be consolidated onto a fewer number of physical machines to reduce power usage and maintenance cost. HotSpots can be handled by increasing the resources available on the physical server or migrating the VM to less loaded physical server. Server consolidation is done with the help of Live virtual machine migration. Performance of live virtual machine migration is measured in total migration time and downtime during migration. The optimization goals of live virtual migration are to minimize total migration time, downtime during which services are totally unavailable to the customer and ensure that migration process does not unnecessarily disrupt active services through resource contention (e.g. CPU, n/w bandwidth) with migrating VM. Also it is important to decide when to trigger the migration, which VM to migrate and where to migrate so that HotSpot is efficiently resolved.

Virtual machine migration process itself takes significant amount of resources like CPU and network resources as it is iteratively doing memory copying work. This can impact the performance of virtual machine under migration as well as application running on other VMs. Many migration performance studies have been found in the literature to predict the performance of migration and tried to avoid overload during migration itself. Generally the triggering of migration is done based on predefined thresholds between 75–80%. VM migration is triggered if any particular resource utilization goes beyond that predefined threshold. The VMs on the physical machine can face the resource deficiency due to contention and may lead to performance degradation if there is rapid increase in resource demand. If the threshold value is high then above scenario occurs and if less for safer side then resources may be underutilized. Hence it is vital to predict the early detection of overloads and then triggering migration than doing prediction to avoid overload during the migration itself. The former is a “real” application overload that needs to be addressed, while the latter is a transient problem due to the migration process.

To address the early detection of application overloads using prediction for triggering VM migration, this paper presents an intelligent VM migration triggering mechanism which uses support vector regression for future workload prediction. In addition to this a model of actual migration process based on pre-copy mechanism is developed and migration performance parameters are evaluated. These parameters are also considered in triggering the decision, which identifies which VM to be migrated for better efficiency. With this, VM migration is triggered if it is predicted that future load is increasing and it is going to exceed the maximum threshold. Thus at the time of migration sufficient resources are available for all the processes and at other time resources are utilized efficiently due to maximum threshold.

The rest of the paper is organised as follows: Section 2 deals with literature survey; Section 3 presents prediction system architecture and design; Section 4 presents resource monitoring system; Section 5 presents overload/hotspot detection system; Section 6 evaluate the system with experiments and results; finally Section 7 presents the conclusion and future work.

2. Literature Review

Christopher Clark *et al.*[1] demonstrated migration of an entire OS instance on a cluster with impressive performance (service downtime as low as 60 ms). They introduced the concept of writable working set (WWS) and present the design, implementation and evaluation of high performance VM migration built on the top of Xen VMM. Migration performance evaluation is made on the page dirty rate and network bandwidth parameters. But the work did not use prediction model for performance evaluation and triggering of migration is based on threshold. Sherif Akoush *et al.*[3] characterised that the parameters affecting live migration are migration link bandwidth and page dirty rate. They presented a AVG (Average Page Dirty Rate) simulation model to predict the pre-copy migration performance giving all the parameters affecting it. It is assumed that page dirty record trace will approximately represent the application. Page dirty trace collected from xen shadow page tables are used to simulate the model. The outlined models are useful for planning and effectively schedule migrations. William Voorsluys *et al.*[4] studied the effects of live migration on performance of application running inside Xen VMs. They evaluated the performance degradation of Web 2.0 application running inside VMs while they were being migrated. The objective they concerned is the SLA violation rather than performance of migration. If less concurrent users then most stringent SLA can be met. Haikun Liu *et al.*[5] presented a model to quantitatively predict the migration performance(migration latency, downtime and network traffic generated) and energy cost of each VM migration. It is based on statistical data analysis of data from historical migration traces on xen platform. Energy consumption is proportional to network traffic generated due to VM migration.

Previous work has also applied neural networks and stochastic models like autoregressive filters to predict the resource demand. John Prevost *et al.*[6] applied multi-layer neural network perceptrons to predict the future load of applications in cloud data centers. The load is predicted by training the perceptrons in a supervised manner with backpropogation algorithm. It is computationally efficient than control theory or reinforcement learning as prediction is feed-forward but training process can be computationally expensive. It can be guaranteed to convergence to an optimal solution. The auto-regressive family of predictors are also used by Sandpiper[7,8], where AR(n) uses n prior observations to make a prediction. This is computationally efficient for low order models. The convergence is guaranteed and minimizes mean square error. But it is not suitable for high order models. A. Ganapthi *et al.*[9] used machine learning approach (Kernel Canonical Correlation Analysis Algorithm) to predict multiple performance metric of both long and short- running database queries. In addition to these techniques trace based resource management takes various resource management decisions by considering historical application resource demands. Rolia *et al.*[10] calculated burst factor to perform dynamic resource allocation. The most recent resource demands are multiplied by burst factor to predict future resource demand directly.

From the literature survey, it is observed optimization solution have been suggested to reduce that downtime and total migration time. Also performance study has been carried out by predicting the VM migration performance. The triggering of VM migration is executed with the help of predefined thresholds which are sufficiently high. Hence load prediction is necessary for early detection of overloads and accordingly trigger the migration beforehand (before crossing the threshold). Thus there will be enough resources for migration process and applications running on the physical server.

3. System Architecture

The proposed system consist of central controller to which all the physical machines are connected. Every physical machines are with Xen hypervisor having multiple virtual machines running on it. Resource monitoring system is installed in Dom-0 of every physical machine which collects resource utilization of every virtual machines at regular interval and send it to resource usage collector in controller. Dom-0 monitoring system collects resource statistics such as CPU, memory and network bandwidth usage. The central controller consist of three components which are resource usage collector, hotspot detector and migration manager as shown in Fig. 1.

The resource usage collector collects the usage of every VM on every physical machine. The monitoring system of every physical machine send the resource usage to the usage collector. It adds the usage of all VMs of every physical machine and store it as resource usage of a physical machine. Hotspot detector detects the hotspot by looking at current resource usage of a physical machine and predicted workload. The Migration manager is responsible for triggering actual migration of VM. It finds out the proper VM to be migrated and where to be migrated.

4. Resource Monitoring System

The management domain i.e. Dom-0 of every Xen hypervisor contains the resource monitoring system which captures CPU, Memory and Network bandwidth utilized. This system collects and sends reports to Resource Usage Collector in the time interval of 10 sec. The system stored 300 such usage reports of every resources.

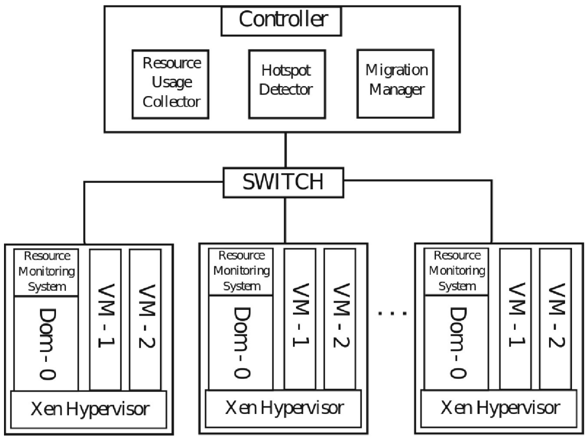

Fig. 1. Dynamic resource provisioning system design.

4.1 CPU monitoring

Dom-0 of Xen Virtual Machine Monitor (VMM) captures the CPU scheduling events. It is tracking the time when the VM is scheduled to get CPU and when it is released from it. With this CPU resource utilization of all the VMs running on the physical machine is captured. Dom-0 processes the I/O and Networking requests on behalf of VMs. The CPU requirement for this is charged on individual VM depending on how many requests served for that VM. This CPU monitoring is done with 10 sec time interval. Xenmon tool with Xen works on above specified principle. It is modified to get the required results.

4.2 Network monitoring

Dom-0 in Xen implements Virtual Firewall Interface (VFR)[1] interface which other domains access through clean device abstraction. Each VM attaches one or more virtual interfaces to VFR. The networking statistics is gathered in /proc/net/dev file which can be accessed and the network utilization is captured in the specified time interval which is configured to 10 sec in our case.

4.3 Memory monitoring

Xen maintains shadow page tables[7] to track memory accesses of guest VMs. It is translated from guest page table on demand. It is specifically used by virtual machine migration algorithm to determine which pages are dirtied during migration. But trapping each memory access adds significant overhead. Hence the memory usage is inferred as memory pressure exists or not by tracking number of read/write to swap partitions on NFS disk. Xentop tool is used to get the memory usage at 10 sec interval.

These readings are stored in two ways. First, time series of individual resource usage of individual VM, collective load of individual resource of all VMs on one PM. Second, probability distribution of usage, in which histogram of all observed usage of individual resource within a specific interval is computed which is normalized to get this distribution.

5. Overload/Hotspot Detection

If the resource usage goes beyond the safety threshold which is 50% in our case then the further workload is predicted with the help of time series prediction with *Auto-regressive models* and *support vector regression* (SVR). The time series prediction[11] can be mathematically stated as

$$\hat{x}(t + \Delta t) = f(x(t - a), x(t - b), x(t - c), \dots) \qquad (1)$$

where $x(t + \Delta t)$ is predicted value of discrete tie series x value. The objective of time series prediction is to find a function $f(x)$ such that the predicted value at a future point at time t is unbiased and consistent. By using regression analysis following equations defines prediction function for linear and non-linear data.

$$f(x) = (w \cdot x) + b \tag{2}$$

$$f(x) = (w \cdot \phi(x)) + b. \tag{3}$$

If the data is non-linear in its input space, it is necessary to map the data $x(t)$ to a higher dimension "feature" space via this kernel function given in equation (3) and then perform a linear regression in high dimensional mapped values. The goal is to find optimal weights w and threshold b.

5.1 Autoregressive modelling and prediction

Mathematically the *Auto Regressive* $AR(p)$ model[12] can be expressed as

$$x(n) = \sum_{i=1}^{p} a(i) * x(n-i) + e(n) \tag{4}$$

where $x(n)$ is predicted linearly from its past p values $x(n-1), x(n-2), \ldots, x(n-p)$. The variable p is called as order of prediction and model is called as *p-order AR* model. Prediction coefficients or model parameters $a(i)$s are calculated with the help of Yule-Walker Method. $e(n)$ is the mean error term.

5.2 Support vector regression

The goal of Support Vector Machine (SVM) is to minimize the regularized risk which is given by equation (5)

$$R_r(f) = R_e(f) + \frac{\lambda}{2}\|w\|^2 \tag{5}$$

where

$$R_e(f) = \frac{1}{N}\sum_{i=o}^{N-1} L(x(i), y(i), f(x(i), w)) \tag{6}$$

and The scaling factor λ is called as regularization constant.

$L(.)$ is loss or cost function to be defined. By using Least Square SVM the goal is to minimize equation (7)

$$\frac{1}{2}\|w\|^2 + C\sum_{i=o}^{N-1} L(x(i), y(i), f(x(i), w)) \tag{7}$$

where

$$L(x(i), y(i), f(x(i), w)) = \begin{cases} |y(i) - f(x(i), w)| - \epsilon, & \text{if } |y(i) - f(x(i), w)| \geq \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

The constant C in equation (7) includes the $(1/N)$ summation normalization factor and ϵ is the tube size, referring to the precision by which the function is to be approximated. Solving for the optimal weights and bias values is an exercise in convex optimization, which is made much simpler by using Lagrange multipliers and forming the dual optimization problem given by (9):

Maximize:

$$-\frac{1}{2}\sum_{i,j=1}^{N}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x(i), x(j)\rangle - \epsilon\sum_{i=1}^{N}(\alpha_i - \alpha_i^*) + \sum_{i=1}^{N} y(i)(\alpha_i - \alpha_i^*) \tag{9}$$

subject to:

$$\sum_{i-1}^{N}(\alpha_i - \alpha_i^*) = 0 : \alpha_i, \alpha_i^* \in [0, C] \tag{10}$$

The approximation of function $f(x)$ is given by the sum of optimal weights times the dot products between the data points as

$$f(x) = \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)\langle x, x(i)\rangle + b \tag{11}$$

To carry out non-linear regression using SVR it is necessary to map input space into feature space with following kernel function

$$k(x, x') = \langle \phi(x), \phi(x')\rangle \tag{12}$$

which can be directly put in equation (11) and weights can be computed in feature space.

5.3 Migration manager

If hotspot is detected by the hotspot detection module then migration manager will find out which overloaded VM is to be selected for migration and where to be migrated. It selects the VM such that migration overhead will be less and total migration time will be less. VM can be overloaded on one or many of the three resource utilization readings i.e. CPU, memory and network traffic. We will unify these readings in one metric called "load" as shown below

$$\text{load} = \frac{1}{1 - cpu} * \frac{1}{1 - nw} * \frac{1}{1 - mem} \tag{13}$$

where cpu, nw and mem are utilization readings of CPU, network and memory normalized by max no. CPUs, max network interfaces and max amount of memory allocated to VM respectively.

The model for actual migration process to predict the performance is described below. VM with smaller size of memory image and smaller λ (ratio of memory dirty rate and memory transmission rate) would lead to less network traffic and shorter migration latency, hence it should be the better candidate for migration[5]. The parameters needed for modelling are VM size, workload memory dirty rate, N/w transmission rate[3]. All these are input parameters and it is necessary to determine memory dirtying rate of an application. Every workload is having set of hot pages which are dirtied again and again and are skipped during the iterative pre-copying procedure. This set of pages is called as Writable Working Set (WWS). The memory dirtying rate can be calculated with dirty bit map using shadow page tables[1]. Generally writable working set of most of the applications is approximately proportional to the pages dirtied in each precopying round[13]. Hence,

$$WWS_i = \eta \cdot V_i \tag{14}$$

where η is a variable correlating time duration of each iteration T_i and memory dirtying rate mdr.

$$\eta = \alpha \cdot T_{i-1} + \beta \cdot mdr + \gamma \tag{15}$$

For a particular application multiple observations are taken for the number of pages dirtied in each iteration and time duration of respective iteration to calculate η. By solvings these simultaneous linear equations model parameters α, β and γ are calculated. The model for entire migration process is given in Algorithm 1. It iterates through the memory copying process and come out when stop and copy conditions satisfied. Finally it gives V_{mig} (total network traffic generated during migration process) and T_{mig} (total migration time required). Algorithm 2 gives the method for selecting highly loaded VM from highly loaded server so that V_{mig} and T_{mig} are less. Following this algorithm triggering of suitable VM is initiated. Algorithm 2 sorts the physical servers in decreasing order of their loads. Within each PM, VMs are assumed to be ordered in decreasing order of load/VM_{size}. It considers first highest load/VM_{size} valued VMs for which V_{mig} and T_{mig} are less. Then it is checked for whether it can be shifted to least loaded PM. It can be done if sufficient resources are available on the destination PM. If not then next least loaded server is

```
Input: $VM_{size}$, $V_{thr}$, $mdr$, $mtr$, $MAXROUNDS$.
Output: C, $T_{mig}$.
let $V_o = VM_{size}$, $i = 0$ ;
repeat
    $T_i = V_i / mtr$;
    $\eta = \alpha\ T_{i-1} + \beta.\ mdr + \gamma$ ;
    $WWS_{i+1} = \eta.T_{i-1}.\ mdr$ ;
    $V_{i+1} = T_i.\ mdr - WWS_{i+1}$ ;
    if $V_{i+1} \leq V_{thr}$ OR $V_{i+1} > p{*}VM_{size}$ then
        $V_{i+1} = T_i.\ mdr$ ;
        $T_{i+1} = V_{i+1} / mtr$ ;
        $T_{down} = T_{i+1} + T_{resume}$;
        break;
    end
    $i = i + 1$;
until $i < MAXROUNDS$;
$V_{mig} = \sum_{i=o}^{MAXROUNDS} V_i$ ;
$T_{mig} = \sum_{i=o}^{MAXROUNDS} T_i$;
```

Algorithm 1. Model of VM migration

```
Input: Load on individual VM on every PM, Total load on PM
Output: VM to be migrated, Destination PM where seleced VM will be migrated
Within each PM arrange the VMs in the decreasing order of $load / VM_{size}$;
repeat
    Select the VM from highest loaded server having highest $load/VM_{size}$ value and minimum $V_{mig}$ and $T_{mig}$
    values;
    repeat
        Select the next least loaded PM as a destination for candidate VM;
    until Least loaded server found with sufficient resources available;
    if No PM can satisfy the candidate VM then
        Select the VM from highest loaded server having next highest $load/VM_{size}$ value and minimum $V_{mig}$ and
        $T_{mig}$ values;
    end
until Utilization of all resources fall under threshold;
```

Algorithm 2. Selection of VM for migration

checked for the same. This is repeated until match is found for candidate VM. If no suitable PM is found then next highest load/VM$_{size}$ VM is selected for which V_{mig} and T_{mig} are less. This procedure is repeated until utilization of all resources falls below thresholds. The Algorithm 2 next consider the next highly loaded PM which is experiencing hot-spot and repeats the above process until all PMs left with no hot-spot.

6. Experimental Setup and Result Analysis

6.1 System description

The experimental setup consist of four Intel Core i7 machines with 4 GB RAM and connected with gigabit ethernet. All these runs Linux 12.04 Server LTS with Xen 4.1.4. The VM images are stored on shared drive created with NFS (Network File System) Server. Each VM on PM runs specific softwares application to create workload. These softwares are selected so as to create different types of workloads such as CPU intensive, memory intensive and network intensive workloads. Also these applications vary in their page dirtying rate. Every PM runs four VMs with different applications running on it. Following softwares applications are used to generate load on VMs – (1) RUBiS[14] – It is an auction site prototype used to evaluate application design patterns and application servers performance scalability. PHP implementation of RUBiS web server and MySQL database server are installed on different VMs, (2) File Server – It is a vsftpd file transfer protocol storing files of different sizes. Client requests are generated to download these files

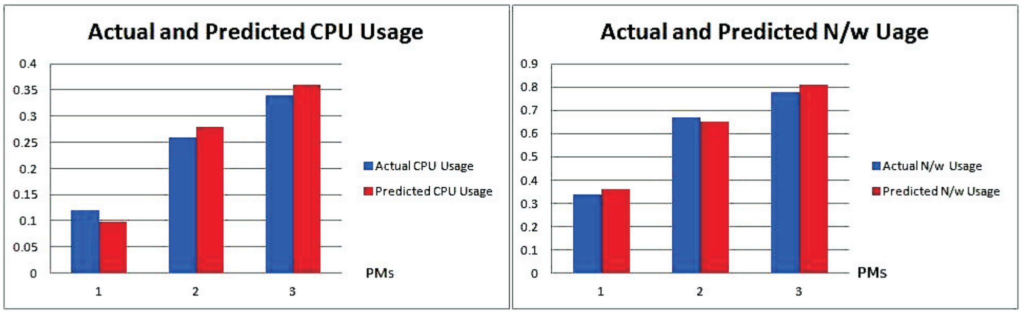

Fig. 2. Actual and predicted CPU and network utilization.

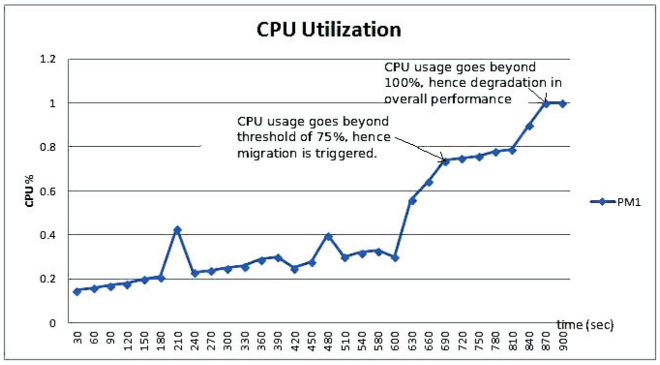

Fig. 3. CPU utilization of PM1 with triggering based on threshold.

which will create a network workload, (3) CPU intensive PHP Scripts – The PHP scripts are written in such a way that these will places a large CPU load on server with low client request rate, (4) httperf – It provides a flexible facility for generating various HTTP requests with different rates and for measuring server performance.

6.2 Actual and predicted CPU and network usage

Time series prediction is done with both AR-p model prediction algorithm and with SVR (Support vector regression). The implementation of these is carried out with machine learning Python Library (scikit-learn). The time series values for all resources are given to the prediction module and future values at an interval of 120,180,240 sec are predicted. The actual values and predicted values are compared. The prediction performance for these intervals are found suitable with SVR. Figure 2 shows the actual and predicted CPU and Network usage values by the SVR prediction algorithm at time interval of 180 sec. The values shows that it has got decent MSE (mean squared error value) of 0.026. With AR-p model we have got MSE of 0.184 for 150 sec interval. But for smaller duration of 10 sec i.e. next value in time series (First Order Prediction) and 30 sec we got very good MSE of 0.0427 and 0.0761 respectively. But it is necessary to predict the value at a large interval, it is decided to predict the values with SVR.

6.3 Resource provisioning with predefined thresholds

When the migration is triggered with predefined thresholds, the performance degradation may occur which is shown by following experiment. As shown in Fig. 3 CPU resource usage was initially low on PM1. CPU intensive workload is given at time 600 sec by running CPU intensive PHP script on VM1. When CPU usage goes beyond predefined threshold of 75%, hot-spot is detected and migration is triggered. But now CPU usage is too high due to CPU usage by other VMs and it becomes contention period where all are in need of CPU. Hence this migrations adds overhead on

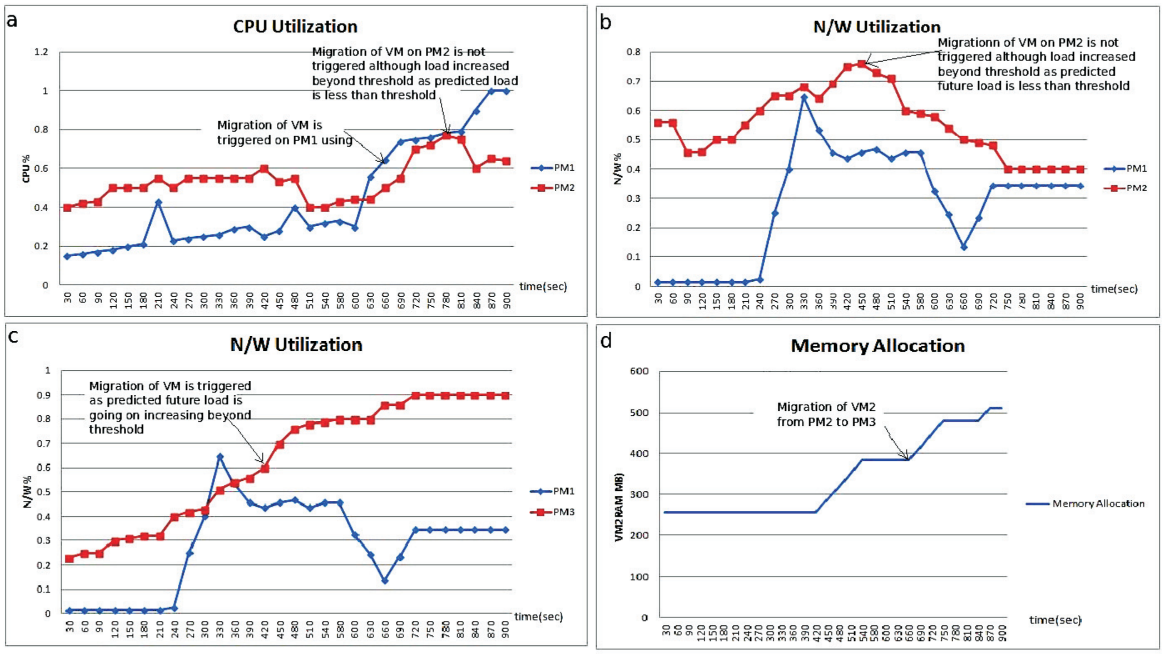

Fig. 4. (a) cpu utilization of PM1 and PM2; (b) network utilization; (c) network utilization of PM1 and PM3; (d) memory allocation for VM2 on PM2.

other applications running on other VMs on this PM. Figure 3 shows that CPU usage reached to almost 100%. Here in this case performance degradation is observed after time 810 sec.

In case of Network utilization, when network usage goes beyond predefined threshold of 75% migration is triggered. But if network utilization further increases because of other VMs as well as migration process itself then there will be contention for network resources and we observed the performance degradation on PM3 as shown in Fig. 4c.

6.4 Resource provisioning with predictions

When CPU resource goes beyond 50%, system predicts the future load and it concludes that further load is increasing drastically at time $t = 630$ sec as shown in Fig. 4a. When usage goes around 60% then migration is triggered at time 660 sec. Hence CPU availability is more for contention as compared to previous case where migration is triggered at a point where CPU usage reached to 75%.

In case of Network Utilization as shown in Fig. 4b, when the network load goes beyond 60%, it is predicted that the network load will increase upto time 450 sec and later it will decrease. Hence at this point migration is not triggered. Thus unnecessary triggering of migrations due to small spikes of workload is also resolved. Figure 4c shows that there is an increase in network utilization from time 400 sec when the resource usage is above 50%. The system predict that future usage is increasing, hence migration for VM on PM3 is triggered at $t = 430$ sec. Hence resource availability in both cases of CPU and Network increases by 15–20% during the case of high load which triggers the migration.

The overall time required to predict the hot-spot and start migration is around 30 sec. In case of memory utilization, if the allocated memory is not sufficient then it is allocated in steps of 32 MB as shown in Fig. 4d. If memory is not enough on that PM then the VM is migrated to other suitable PM. Here the migration is triggered when memory demand reaches to 382 MB.

7. Conclusion

This paper presented an intelligent system which predicts future workload for early detection of overloads so as to trigger migrations in such a way that possible system performance degradation is resolved. Also the actual migrating process of live virtual machine migration is modelled which determines the total network traffic generated and total time required for migration. The migration is triggered based on these predictions and determined performance values instead of triggering the migration based on predefined thresholds. The prediction accuracy received is decent. This helps in resolving the degradation in the system due to overload on PM during migration by increasing the resource availability through early triggering decision. It also removes unnecessary migrations which exists because of small transient spikes in resource usage.

References

[1] Christopher Clark, 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*, NSDI'05, Berkeley, CA, USA, USENIX Association, vol. 2, pp. 273–286, (2005).

[2] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt and Andrew Warfield, Xen and the Art of Virtualization, *SIGOPS Oper. Syst. Rev.*, vol. 37(5), pp. 164–177, October (2003).

[3] S. Akoush, R. Sohan, A. Rice, A. W. Moore and A. Hopper, Predicting the Performance of Virtual Machine Migration, In *International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2010, IEEE*, pp. 37–46, August (2010).

[4] William Voorsluys, James Broberg, Srikumar Venugopal and Rajkumar Buyya, Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation, In *Proceedings of the 1st International Conference on Cloud Computing*, CloudCom'09, Berlin, Heidelberg, Springer-Verlag, pp. 254–265, (2009).

[5] Haikun Liu, Cheng-Zhong Xu, Hai Jin, Jiayu Gong and Xiaofei Liao, Performance and Energy Modeling for Live Migration of Virtual Machines, In *Proceedings of the 20th International Symposium on High Performance Distributed Computing*, HPDC'11, New York, NY, USA, ACM, pp. 171–182, (2011).

[6] John J. Prevost, KranthiManoj Nagothu, B. Kelley and M. Jamshidi, Prediction of Cloud Data Center Networks Loads using Stochastic and Neural Models, In *6th International Conference on System of Systems Engineering (SoSE), 2011*, pp. 276–281, June (2011).

[7] Timothy Wood, Prashant Shenoy, Arun Venkataramani and Mazin Yousif, Sandpiper: Black-Box and Gray-Box Resource Management for Virtual Machines, *Computer Networks*, Virtualized Data Centers, vol. 53(17), pp. 2923–2938, (2009).

[8] Timothy Wood, Prashant Shenoy, Arun Venkataramani and Mazin Yousif, Black-Box and Gray-Box Strategies for Virtual Machine Migration, In *Proceedings of the 4th USENIX Conference on Networked Systems Design &#38; Implementation*, NSDI'07, Berkeley, CA, USA, USENIX Association, pp. 17–17, (2007).

[9] Archana Ganapathi, Harumi Kuno, Umeshwar Dayal, Janet L. Wiener, Armando Fox, Michael Jordan and David Patterson, Predicting Multiple Metrics for Queries: Better Decisions Enabled by Machine Learning, In *Proceedings of the 2009 IEEE International Conference on Data Engineering*, ICDE'09, Washington, DC, USA, IEEE Computer Society, pp. 592–603, 2009.

[10] D. Gmach, J. Rolia, L. Cherkasova and A. Kemper, Capacity Management and Demand Prediction for Next Generation Data Centers, In *International Conference on Web Services, 2007. ICWS 2007, IEEE*, pp. 43–50, July (2007).

[11] N. I. Sapankevych and Ravi Sankar, Time Series Prediction using Support Vector Machines: A Survey, *Computational Intelligence Magazine, IEEE*, vol. 4(2), pp. 24–38, May (2009).

[12] Paola Bermolen and Dario Rossi, Support Vector Regression for Link Load Prediction, *Computer Networks*, QoS Aspects in Next-Generation Networks, vol. 53(2), pp. 191–201, (2009).

[13] Dawei Huang, Deshi Ye, Qinming He, Jianhai Chen and Kejiang Ye, Virt-lm: A Benchmark for Live Migration of Virtual Machine, In *Proceedings of the 2Nd ACM/SPEC International Conference on Performance Engineering*, ICPE'11, New York, NY, USA, ACM, pp. 307–316, (2011).

[14] RUBiS: Rice University Bidding System. http://rubis.ow2.org.