Performance Analysis during Live Migration

Shreya R[1], Shruthi R[2], Anala M.R[3],Shobha G [4]

[1,2] 8th Semester, Bachelor of Engineering
[3] Assistant Professor
[4] Dean PG Studies

Department of Computer Science and Engineering, R.V. College of Engineering, Bangalore.

***Abstract*—The term virtualization describes the separation of a resource or request for a service from the underlying physical delivery of the service. Live migration of virtual machine relocates the memory and virtual device state of a VM from one physical machine to another with no noticeable downtime of the VM. The aim of this paper is to analyze the performance of the chosen workloads during live migration. The measurement and statistics of downtime and total migration time rate which are the key considerations for choosing a live migration approach will be recorded for drawing inferences. The workloads chosen will be intensive tasks on the system in terms of CPU utilization. Inference drawn from this analysis would be to propose which virtual machine should be migrated when the chosen workloads are running simultaneously on each virtual machine and CPU usage by all the running virtual machines exceeds a certain threshold.**

I.INTRODUCTION

Virtualization is the abstraction of computing resources that masks the physical nature and boundaries of resources from resource users to simplify the way in which other systems, applications, or end users interact with those resources.Virtualization allows multiple operating system instances to run concurrently on a single computer. It is a means of separating hardware from a single operating system. Each guest OS is managed by a Virtual Machine Monitor (VMM), also known as a hypervisor.[1]

The objectives of the paper is 1) Configuration and Installation of the open source software for virtualization, 2) To perform live migration of VMs using different iSCSI and 3) To analyze the live migration of virtual different CPU intensive worloads.

The paper is organised as follows Section II is discusses the various Desktop Virtualization Approaches.Section III is about the different live migration approaches, advantages of live migration and the need for it. Section IV is about Internet Small Computer System Interface(iSCSI) which is a set of standards for physical connection and transferring data between computers and peripheral devices. Section V outlines the details of the experimental setup which includes the functional ,hardware ,software requirements,design considerations, programming languages and libraries used , modules of the code of the test suite. Section VI deals with the results and analysis which explains the details of the experiments that were conducted. Section VII is Conclusion which gives the outcome of the work carried out and future enhancements.

II. VIRTUALIZATION APPROACHES

A. *KVM*

KVM (for Kernel-based Virtual Machine) is a full virtualization solution for Linux onx86 hardware containing virtualization extensions (Intel VT or AMD-V). It consists of a loadable kernel module, kvm.ko, that provides the core virtualization infrastructure and a processor specific module, kvm-intel.ko or kvm-amd.ko.

B. *XEN*

Xen is an open-source bare metal hypervisor, which makes it possible to run many instances of an operating system or indeed different operating systems in parallel on a single machine. Key features of Xen are : small footprint and interface, operating system agnostic, driver isolation and para-virtualization.[2]

III. *LIVE MIGRATION APPROACHES*

A. *Pre-Copy Approach*

- Warm-up phase

In memory migration of a VM, usually the Hypervisor copies all the memory pages from source to destination while the VM is still running on the source. If some memory pages change during memory copy process—dirty pages, they will be re-copied until the rate of recopied pages is not less than page dirtying rate.

- Stop-and-copy phase

After warm-up phase, the VM will be stopped in source and the remaining dirty pages will be copied to the destination and VM will be resumed in destination. The time between stopping VM on source and resuming it on destination is called “down-time”.

B. *Post Copy Approach*

Post-copy VM migration is initiated by suspending the VM at the source. With the VM suspended, a minimal execution state of the VM (CPU, registers, and non-pageable memory) is transferred to the target. The VM is then resumed at the target, even though the entire memory state of the VM has not yet been transferred, and still resides at the source. Source host responds to the network fault by sending the faulted page. Since each page fault of the running VM is redirected towards the source, it can degrade the applications running inside the VM. However, when pure demand-paging accompanied with the techniques such as pre-paging can reduce this impact by a great extent.

C. *Advantages of Live Migration*

a) Load balancing - guests can be moved to hosts with lower usage when their host becomes overloaded, or another host is under-utilized.

b) Hardware independence - guests do not experience any downtime for hardware improvements.

c) Energy saving - guests can be redistributed to other hosts and host systems powered off to save energy

d) Geographic migration – guests can be moved to another location for lower latency or in serious circumstances.[3]

D. *Steps in Live Migration*

The logical steps that are followed during the preparation and migration [3] are summarized in Figure 1

a) Stage 0: Pre-Migration: Begin with an active VM on physical host A. To speed any future migration, a target host may be preselected where the resources required to receive migration will be guaranteed.

b) Stage 1: Reservation: A request is issued to migrate an OS from host A to host B. We initially confirm that the necessary resources are available on B

c) Stage 2:Pre-Copy: In this stage, all pages are transferred from A to B.

d) Stage 3: Stop-and-Copy:Suspend the running OS instance at A and redirect its network traffic to B. The copy at A is still considered to be primary and is resumed in case of failure

e) Stage 4: Commitment: Host B indicates to A that it has successfully received a consistent OS image. Host A acknowledges this message as commitment of the migration transaction**.**

f) Stage 5: Activation: The migrated VM on B is now activated. Post-migration code runs to reattach device drivers to the new machine and advertise moved IP addresses.

IV. INTERNET SMALL COMPUTER SYSTEM INTERFACE (iSCSI)

A. *Overview of iSCSI*

iSCSI , stands for Internet Small Computer System Interface, an Internet Protocol based storage networking standard for linking data storage facilities. It carries SCSI commands over IP networks, and is used to facilitate data transfers over intranets, to manage storage over long distances. It can be used to transmit data over local area networks (LANs), wide area networks (WANs), or the Internet and can enable location-independent data storage and retrieval. The protocol allows clients (called initiators) to send SCSI commands (CDBs) to SCSI storage devices (targets) on remote servers. It uses TCP and allows two hosts to negotiate and exchange SCSI commands using IP networks. It takes a popular high-performance local storage bus and emulates it over wide-area networks, creating a storage area network (SAN). It requires no dedicated cabling and can be run over existing IP infrastructure. One of the main requirements of using iSCSI is the configuration of the initiator, target and destination machines which was one of the most challenging parts of the experimental setup.

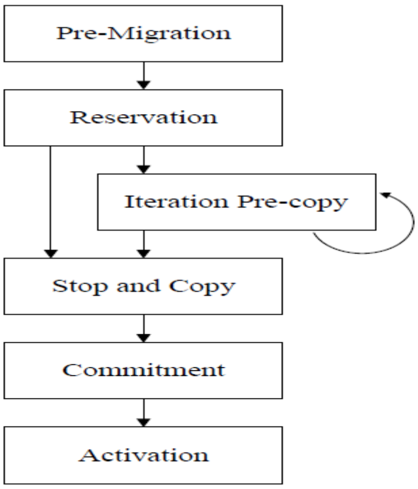


Fig 1:Steps in Live Migration

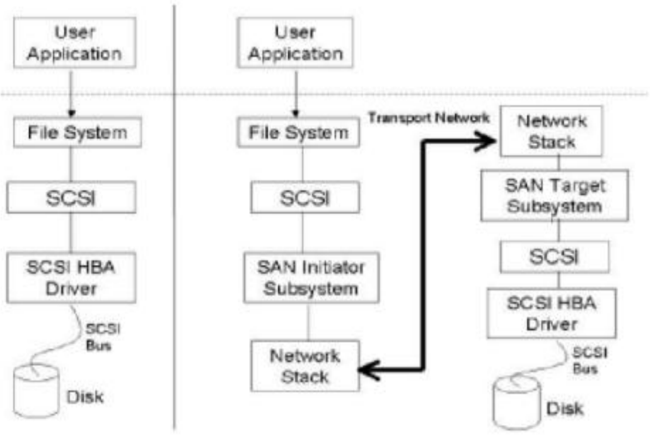


Fig 2 iSCSI: A mapping of SCSI over TCP protocol

B. *Concepts in iSCSI*

Connection is the communication with target occurs over one or more TCP connections. The TCP connections carry control messages, SCSI commands, parameters, and data within iSCSI Protocol Data Units. Session is the group of TCP connections that link an initiator with a session (loosely equivalent to a SCSI I-T nexus). TCP connections can be added and removed from a session. Across all connections within a session, an initiator sees one and the same target.

V. EXPERIMENTAL SETUP

A. *Hardware Requirements*

Processor: An Intel x86 processor with atleast 2MHz clock frequency supporting Intel-VT, the hardware support for virtualization Memory: 4GB DDR3 RAM.Harddisk: 500GB. Connectivity: A 1GBps Ethernet Interface.

B. *Software Requirements*

Fedora is an RPM-based, general purpose collection of software, including an operating system based on the Linux kernel, developed by the community-supported Fedora Project and owned by Red Hat.

C. *Benchmarks*

- Livermore Kernels (Livermore Loops)

This supercomputer benchmark was first introduced in 1970, initially comprising 14 kernels of numerical application, written in Fortran. The number of kernels was increased to 24 in the 1980's. Performance measurements are in terms of Millions of Floating Point Operations Per Second or MFLOPS. The program also checks the results for computational accuracy. One main aim was to avoid producing single number performance comparisons, the 24 kernels being executed three times at different Do-loop spans to produce short, medium and long vector performance measurements.

- Dhrystone Benchmarks

The Dhrystone "C" benchmark, a sort of Whetstone without floating point, became the key standard benchmark, from 1984, with the growth of Unix systems. The first version was produced by Reinhold P. Weicker in ADA and translated to "C" by Rick Richardson. Two versions are available Dhrystone versions 1.1 and 2.1. The second version was produced to avoid over-optimization problems encountered with version 1.

- Linpack Benchmark

This benchmark was produced by Jack Dongarra from the "LINPACK" package of linear algebra routines. It became the primary benchmark for scientific applications from the mid 1980's with a slant towards supercomputer performance. The pre-compiled versions are double precision, rolled. Other versions are available with different sizes of matrices. Performance rating is in terms of MFLOPS.

D. *Design Considerations*

Both the source and destination hosts of the migration should have the same hardware configuration. This is to ensure that the migrated virtual machine can run suitably on the destination. Though the aim is to achieve the live migration of virtual machines with little downtime and total migration time, the primary focus is to analyze the performance of CPU intensive applications during live migration.

E. *General Constraints*

- Software Environment: The VMM must be installed in both source and destination hosts.
- End User Environment: The module created can be executed only by users with root access on the bash terminal.
- Availability of Resources: The destination host must have same configuration of the source host to achieve a successful migration.
- Interoperability requirements: For full virtualization to be effective, the virtualized hardware presented in the guest OS must resemble physical hardware extremely close. The Xen kernel chosen while booting on the hosts should be the same.
- Network communications: Bridged networking allows the virtual interfaces to connect to the outside network through the physical interface, making them appear as normal hosts to the rest of the network. Bridge br0 is created and need to be configured in etc/network/interfaces and bridged with physical NIC.

VI. RESULTS AND ANALYSIS

This section with the details of the experiments that were conducted and demonstrates the performance analysis of CPU intensive workloads during live migration. The first parameter considered to evaluate the performance is the total migration time [4]. The total migration time is the time taken to migrate the VM from one physical machine to other physical machine. For analyzing the performance virtual machines running workloads, following parameters are considered

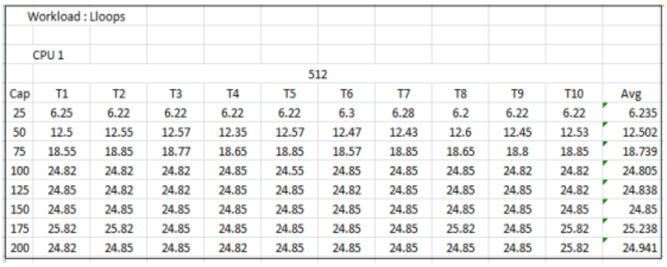
- Memory allocated to each VM
- Number of CPUs allocated to each VM
- Cap value of CPU
- Number of cores allocated to the system during booting

A. *Measuring virtual machine performance*

In order to analyze the performance of the virtual machines running the workloads memory, CPU and cap values are considered. For the purpose of analyses the following workloads are chosen, Dhrystone, Linpack and Livermore Loops. Each VM is running one among the chosen workloads. By allocating one core followed by two cores of the physical machines four cores and in each case by varying the ram value from 512MB to 2048MB and varying the cap value from 25 percent of the CPU to 200 percent the readings are taken noted

- Livermore Loops

Table 1 shows the readings taken when the virtual machine is running the Lloops workload and it is allocated 512MB RAM. Table 2 shows the readings taken when the virtual machine is running the Lloops workload and it is allocated 1GB RAM. Table 3 shows the readings taken when the virtual machine is running the Lloops workload and it is allocated 2GB RAM .The VM is allocated 1CPU.The readings are noted down as the cap value of the cpu is varied from 25 to 200. The above data is plotted as a bar graph as shown in Fig 3

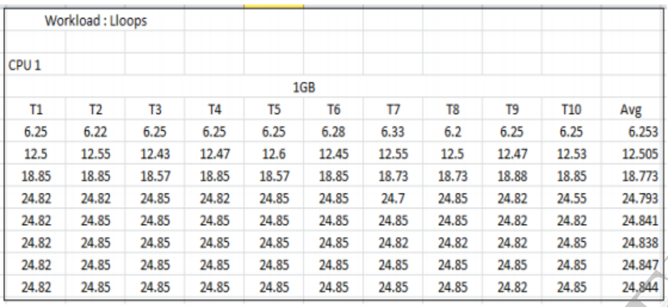


Workload : Lloops

CPU 1

					512						
Cap	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	Avg
25	6.25	6.22	6.22	6.22	6.22	6.3	6.28	6.2	6.22	6.22	6.235
50	12.5	12.55	12.57	12.35	12.57	12.47	12.43	12.6	12.45	12.53	12.502
75	18.55	18.85	18.77	18.65	18.85	18.57	18.85	18.65	18.8	18.85	18.739
100	24.82	24.82	24.82	24.85	24.55	24.85	24.85	24.85	24.82	24.82	24.805
125	24.85	24.82	24.82	24.85	24.85	24.82	24.85	24.85	24.85	24.82	24.838
150	24.85	24.85	24.85	24.85	24.85	24.85	24.85	24.85	24.85	24.85	24.85
175	25.82	25.82	24.85	24.85	24.85	24.85	24.85	25.82	24.85	25.82	25.238
200	24.82	24.85	24.85	24.82	24.85	24.85	24.85	24.85	24.85	25.82	24.941

Table 1: Workload Lloops with 512MB RAM and 1 CPU

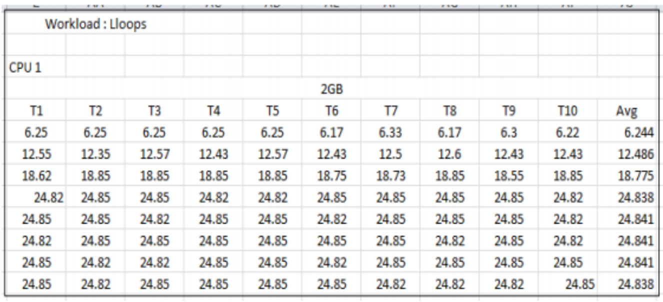


Workload : Lloops

CPU 1

				1GB						
T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	Avg
6.25	6.22	6.25	6.25	6.25	6.28	6.33	6.2	6.25	6.25	6.253
12.5	12.55	12.43	12.47	12.6	12.45	12.55	12.5	12.47	12.53	12.505
18.85	18.85	18.57	18.85	18.57	18.85	18.73	18.73	18.88	18.85	18.773
24.82	24.82	24.85	24.82	24.85	24.85	24.7	24.85	24.82	24.55	24.793
24.82	24.85	24.85	24.85	24.85	24.85	24.85	24.85	24.82	24.82	24.841
24.82	24.85	24.85	24.85	24.85	24.85	24.82	24.82	24.82	24.85	24.838
24.82	24.85	24.85	24.85	24.85	24.85	24.85	24.85	24.85	24.85	24.847
24.82	24.85	24.85	24.85	24.85	24.85	24.85	24.85	24.82	24.85	24.844

Table 2: Workload Lloops with 1GB RAM and 1 CPU



Workload : Lloops

CPU 1

					2GB					
T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	Avg
6.25	6.25	6.25	6.25	6.25	6.17	6.33	6.17	6.3	6.22	6.244
12.55	12.35	12.57	12.43	12.57	12.43	12.5	12.6	12.43	12.43	12.486
18.62	18.85	18.85	18.85	18.85	18.75	18.73	18.85	18.55	18.85	18.775
24.82	24.85	24.85	24.82	24.82	24.85	24.85	24.85	24.85	24.82	24.838
24.85	24.85	24.82	24.85	24.85	24.82	24.85	24.85	24.85	24.82	24.841
24.82	24.85	24.85	24.85	24.85	24.85	24.85	24.82	24.85	24.82	24.841
24.85	24.82	24.82	24.85	24.85	24.82	24.85	24.85	24.85	24.85	24.841
24.85	24.82	24.85	24.85	24.85	24.85	24.82	24.82	24.82	24.85	24.838

Table 3:Workload Lloops with 2GB RAM and 1 CPU

Fig 3 shows the readings taken when the virtual machine is running the Lloops workload and it is allocated 512MB RAM. Fig 4 shows the readings taken when the virtual machine is running the Lloops workload and it is allocated 512MB,1GB and 2GB RAM and 2 CPUs are allocated.

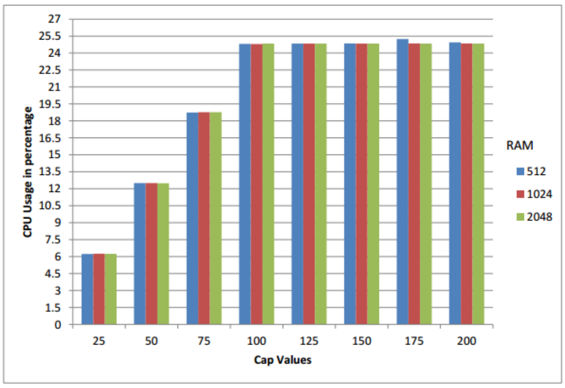


Fig 3: CPU usage by Livermore Loops on VM with 1 CPU

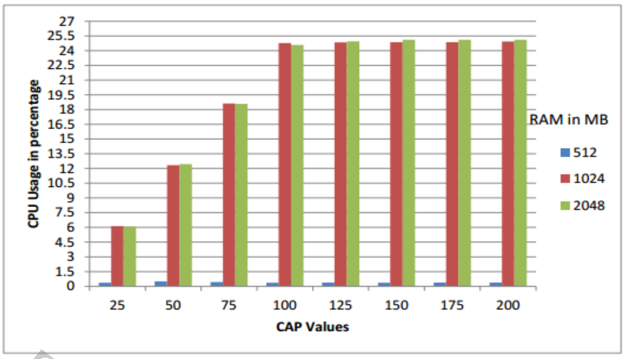


Fig 4: CPU usage by Livermore Loops on VM with 2 CPUs

- Dhrystone

Fig 5 shows the readings taken when the virtual machine is running the Dhrystone workload and it is allocated 512MB,1GB and 2GB RAM and 1 CPU is allocated.Fig 6 shows the readings taken when the virtual machine is running the Dhrystone workload and it is allocated 512MB,1GB and 2GB RAM and 2 CPUs are allocated.

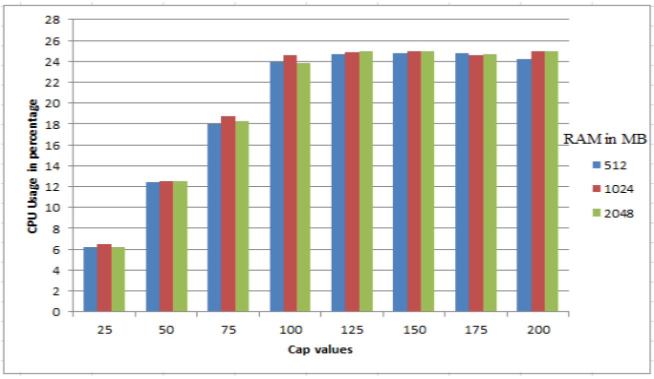


Fig 5 :CPU usage by Dhrystone on VM with 1 CPU

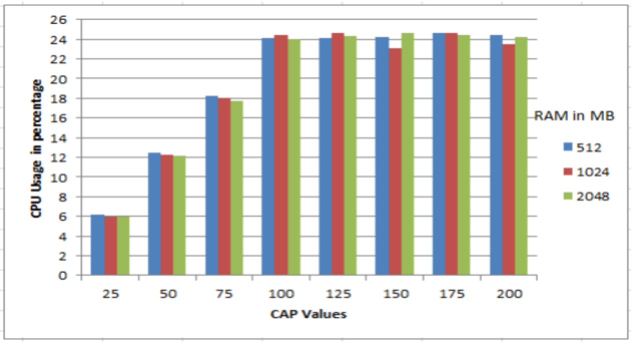


Fig 6 :CPU Usage by Dhrystone on VM with 2 CPUs

- Linpack

Fig 7 shows the readings taken when the virtual machine is running the Linpack workload and it is allocated 512MB,1GB and 2GB RAM and 1 CPU is allocated. Fig 8 shows the readings taken when the virtual machine is running the Linpack workload and it is allocated 512MB,1GB and 2GB RAM and 2 CPUs are allocated.

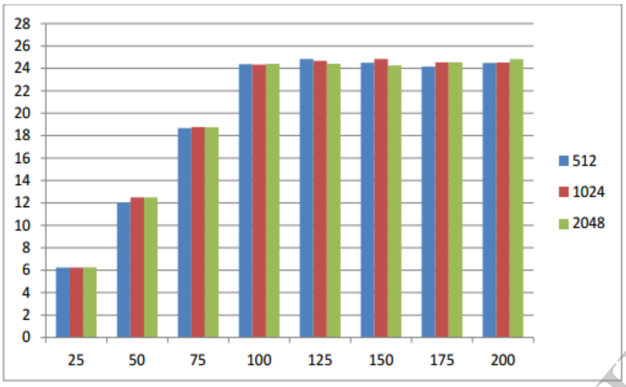


Fig 7 :CPU usage by Linpack on VM with 1 CPU

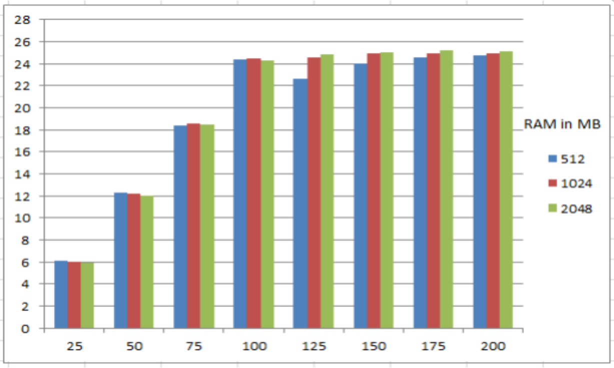
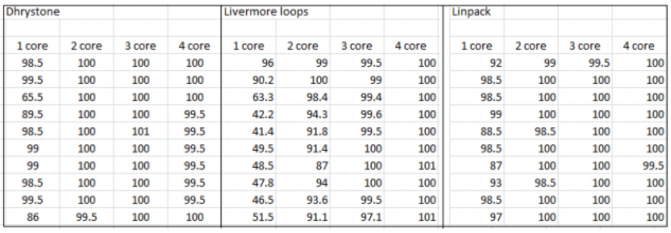


Fig 8 :CPU usage by Linpack on VM with 2 CPUs

B. *Measuring Performance of Physical System*

From table 4 it can be analysed that there is a significant increase in the CPU usage by all the chosen CPU intensive workloads when the number of CPUs or cores allocated to the physical machine is increased from 1 to 2. There is slight increase in CPU usage when number of cores is increased from 2 to 3. When 4 cores or CPUs are allocated the CPU usage over all trials is nearly 100 percent.



Dhrystone				Livermore loops				Linpack			
1 core	2 core	3 core	4 core	1 core	2 core	3 core	4 core	1 core	2 core	3 core	4 core
98.5	100	100	100	96	99	99.5	100	92	99	99.5	100
99.5	100	100	100	90.2	100	99	100	98.5	100	100	100
65.5	100	100	100	63.3	98.4	99.4	100	98.5	100	100	100
89.5	100	100	99.5	42.2	94.3	99.6	100	99	100	100	100
98.5	100	101	99.5	41.4	91.8	99.5	100	88.5	98.5	100	100
99	100	100	99.5	49.5	91.4	100	100	98.5	100	100	100
99	100	100	99.5	48.5	87	100	101	87	100	100	99.5
98.5	100	100	99.5	47.8	94	100	100	93	98.5	100	100
99.5	100	100	99.5	46.5	93.6	99.5	100	98.5	100	100	100
86	99.5	100	100	51.5	91.1	97.1	101	97	100	100	100

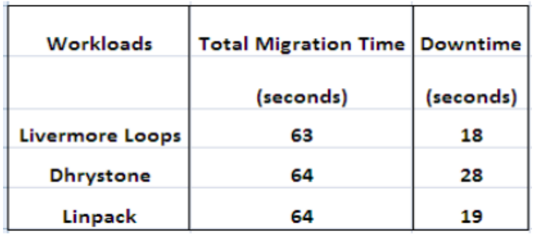
Table 4 :CPU Usage by workloads on physical machine with varying core values

C. *Measuring Performance of Live Migration*

Total Migration Time[3][4] time may be defined as the sum of the time spent on all migration stages from initialization at the source host through to activation at the destination.Total migration time is given by the equation (1). The time taken to migrate virtual machine from one physical machine to another physical machine.

$$TotalMigrationTime = Initialization + Reservation + \sum_i PreCopy_i + Stop\&Copy + Commitment + Activation \qquad (Equation\ 1)$$

Total downtime starts to increase in proportion to the increase in the number of modified pages that need to be transferred in the stop and copy stage. Total downtime further increases until the defined upper bound in which it has to send the entire VM memory. Total migration time also increases with an increasing page dirty rate. This is attributable to the fact that more modified pages have to be sent in each pre-copy round. Moreover, the migration sub-system has to go through more iteration with the hope to have a short final stop and copy phase.



Workloads	Total Migration Time	Downtime
	(seconds)	(seconds)
Livermore Loops	63	18
Dhrystone	64	28
Linpack	64	19

Table 5: Total Migration Time and Downtime of the VMs when running each workload.

The VM to be migrated when CPU usage crosses a certain threshold is the one with the least total migration time, in case total migration time is less the VM with lesser downtime should be migrated. Hence among the three chosen CPU intensive workloads first preference is given to the VM running Livermore loops then Linpack followed by Dhrystone when live migration is initiated to reduce CPU usage by the active domains.

VII. CONCLUSION

Server Administrators face a number of pressing problems while managing processes running on different virtual machines for example load balancing, hardware independence ,energy saving, and geographic migration. The scope of this paper lies in serving as a solution to the above problems by running a script file which observes the performance of two virtual machines that are running on the physical machine. This paper aims at providing a descriptive and detailed measurement of various attributes of computer performance like CPU Utilization, memory utilization, network latency etc.

The outcome of this paper is to propose which VM should be migrated using Xen as the hypervisor when the total CPU usage by the active domains crosses sixty five percent of the total system CPU usage.

VIII. FURTHER STUDY

The analysis can be extended to support the following functionality.

- Support to other architectures: Analysis can be enhanced to support the other architecture of the hypervisors.
- Load Balancing: The current work does not take into consideration of the destination machine hence migration from source to destination and vice versa should be automated to balance load between the two systems.
- Improved pre-copy approach and post-copy approach: Improved pre-copy approach [5] can be used for better performance of VM migration.
- Live migration with authentication: the current work does include authentication during live migration on the source, destination,or target.

IX. REFERENCES

[1]Smith, J.E.;Ravi Nair;Dept. of Electr. & Comput. Eng., Wisconsin Univ., Madison, WI, USA Smith, James E. "The Architecture of Virtual Machines".Computer (IEEE Computer Society),ISSN: 0018-9162, Pages 32-38, Digital Object Identifier:10.1109/MC.2005.173.

[2]Kivity Avi, Kamay Yaniv, Laor Dor, Lublin Uri, Ligouri Anthony, “KVM: the Linux Virtual Machine Monitor”,proceedings of the Linux Symposium Volume one, Ottawa Canada,2007, pages 225-230.

[3]Christopher Clark, Keir Fraser, Steven Hand, Jakob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, Andrew Warfield, JJ Thomson,“Live Migration of Virtual Machines”, NSDI'05 Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2 ,USENIX Association Berkeley, CA, USA 2005.

[4] Akoush, S.; Sohan, R.; Rice, A.; Moore, A.W.; Hopper, A.Computer. Lab., University of Cambridge, Cambridge, UK “Predicting the performance of virtual machine migration”, IEEE International Symposium on: Modeling, Analysis & Simulation ofComputer and Telecommunication Systems (MASCOTS), 2010, pages: 37 – 46, Digital Object Identifier: 10.1109/MASCOTS.2010.13.

[5]Fei Ma; Feng Liu; Zhen Liu Network Manage. Res. Centre, Beijing Jiaotong Univ., Beijing, China “Live virtual machine migration based on improved pre-copy approach ”IEEE International Conference on Software Engineering and Service Sciences (ICSESS), 2010, Print ISBN:978-1-4244-6054-0, page(s):230 – 233, Digital Object Identifier