Juni Khyat
(UGC Care Group I Listed Journal)
ISSN: 2278-4632
Vol-10 Issue-8 No. 2 August 2020

Performance Analysis of KVM and XEN Hypervisor for Resource Monitoring in Cloud Computing

Nitesh Saxena,
M. Tech Scholar,
Poornima University, Jaipur
onlynitesh.smcet@gmail.com

Dr. Madan Lal Saini,
Associate Professor,
Poornima University, Jaipur
madan.saini@poornima.edu.in

ABSRTACT

There are different choices when choosing a hypervisor for different requirements like the performance of the hypervisor, for how long time we are using the technology, how easy to merge with different available platforms, what are the profitable inferences and support for different guests and functions. In our research paper, we tried to help all by making you all able to select the perfect hypervisor among KVM and XEN by testing them out on the basis of various parameters. Optimization of Xen Hypervisor and Optimization of KVM Hypervisor was made possible by carrying out various tests with different loads done during this research work.

Keywords: KVM, VM, HVM, XEN, IOPS.

1. INTORDUCTION

Cloud computing is technology that provides us internet based services on demand and ease of using them from anywhere as these are ubiquitous in nature. The services uses computing stacks from the infrastructure available placed on virtual machines to provide software services.

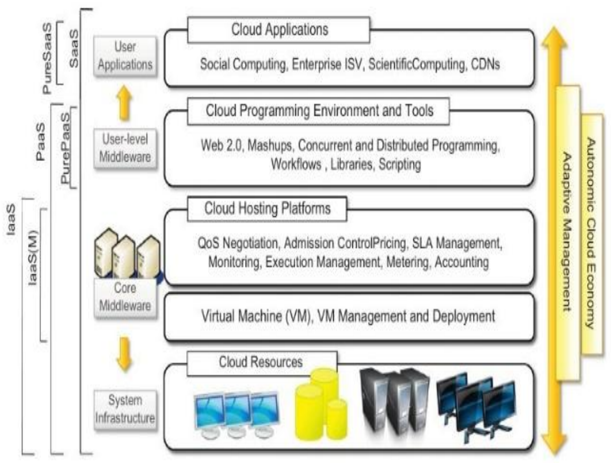

PureSaaS
SaaS
PaaS
PurePaaS
IaaS
IaaS(M)
User Applications
User-level Middleware
Core Middleware
System Infrastructure
Cloud Applications
Social Computing, Enterprise ISV, ScientificComputing, CDNs
Cloud Programming Environment and Tools
Web 2.0, Mashups, Concurrent and Distributed Programming, Workflows , Libraries, Scripting
Cloud Hosting Platforms
QoS Negotiation, Admission ControlPricing, SLA Management, Monitoring, Execution Management, Metering, Accounting
Virtual Machine (VM), VM Management and Deployment
Cloud Resources
Adaptive Management
Autonomic Cloud Economy

Fig 1.1 Architectural Diagram of Cloud Computing

In this work, we have installed Ubuntu on two hypervisors i.e. KVM and XEN. By taking various parameters i.e. Disk I/O, CPU performance, network performance and memory management, we compared the two hypervisors XEN and KVM to analyze their performances. Firstly, we have installed ganglia in our host system to measure the performance of the guest and the host machine. Then we have installed one VM and measured the performance and then we installed another VM to increase the load and checked the performance of the hypervisor. Secondly, we have installed Citrix server and then installed one virtual machine and checked the performance and then installed another VM and measured the performance on the basis of parameters. In the results, we have analyzed the overall performance of XEN and KVM, their performance separation and scalability in a measureable manner and after the comparison we came on to a conclusion that the major difference between these two hypervisors was with the scalability factor. We have seen that KVM used to get crashed when we load 4 or above guests over it which is not the case of XEN. If we talk about kernel compile test then XEN is far ahead of KVM but in case of I/O intensive tasks then KVM takes a clear lead. The KVM's performance is also better during isolation ever so slightly then XEN. We will comparing these two by including Xen with full virtualization (HVM) and KVM with para virtualized I/O.

There are many considerations when selecting a hypervisor such as the performance, how mature the technology is, how it integrates with existing systems, the commercial implications and guest and functionality support. In this section, there's a guide that sets out to help you select the most appropriate hypervisor. Within this section, we compared two major hypervisors – KVM and XEN. Here is a snapshot of the hypervisors and a bit about each:

1.1 **Kernel Virtual Machine:** - A Linux based open source hypervisor. First introduced into the Linux kernel in February 2007, it is now a mature hypervisor and is probably the most widely deployed open source hypervisor in an open source environment. KVM is used in products such as Redhat Enterprise Virtualization (RHEV).

1.2 XEN: Xen provides a mechanism by which a guest operating system can be given direct access to a physical device. The guest OS can then use existing device driver.

1.3 Theoretical Aspects of Targeted Work

Many researchers have used different monitoring of resources, correlation between resources and workload predication in cloud to work out a feasible and acceptable solution using various existing & modified procedure to read, analyze, and evaluate the details. In this chapter we will discuss virtualization techniques, resource monitoring setup and correlation coefficients with reference to the research works carried out by various researchers.

2. VIRTUALIZATION TECHNIQUES

In today computing scenario the process of virtualization represents to make a virtual type of something like virtual platforms of computer hardware, storage devices and network resources. It is a procedure which reasonably divides the resources of system between various processes provided by mainframe computers. The virtualization term represent to execute many operating system on one physical host machine using Xen tool. The optimize approach to implement visualization that is essential to clear understanding of various solution of vitalization which is recently available. The main objective to discuss about general terms of four virtualization procedure commonly use today specifically guest operating system, shared kernel, hypervisor and kernel level.

Guest Operating System Virtualization: It is easiest process of virtualization where typical original operating systems like Linux, UNIX, Windows and MacOS X run on physical host computer system. A virtualization application is executing this operating system in a similar way as other applications like spreadsheet and word processor would execute on host system. The virtualization process basically control access to resource of physical hardware on behalf of each virtual machine and responsible to start, stop and manage individual VM. This virtualization process involves in a binary rewriting process that contains scanning the stream of instruction of running guest system and substituting any privileged instruction with safe simulation. The outcome of creating guest system considers it directly run on hardware system slightly than a virtual machine within an application. Guest operating system virtualization technologies examples are VMware Server and Virtual Box.

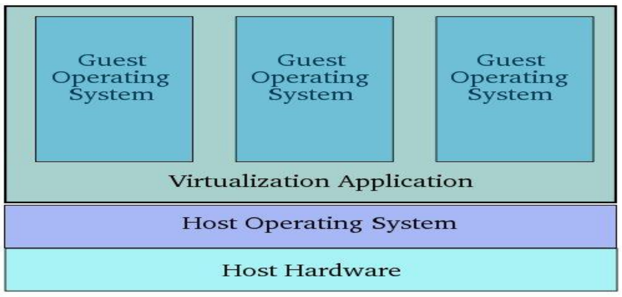


Fig 2.1: General Diagram of Guest Operating System Virtualization

In the figure 2.1 the virtualization process various virtual machines operate guest operating system execute on top of host operating system in similar way as other process. Obviously there are many abstraction layer between the essential host hardware system and guest operating systems are not favorable to performance of high levels of virtual machine. This approach has benefit which is not essential to change either host or guest operating system and no special CPU hardware is needed for virtualization support.

Shared Kernel Virtualization: This type of virtualization also referred to as system level and operating system virtualization also taking the benefits of architectural design of OS based on UNIX and Linux. If you correctly understand the process of shared kernel virtualization it also helps to understand about the two major component of UNIX and Linux operating system. The kernel represent to core part of operating system and simply it handles complete instruction used to occur in between of operating system and physical hardware. The next main component comprises all the libraries, files and utilities represent to root file system which is main function of operating system. Each virtual guest operating system in shared kernel virtualization contain their own root file system but share the similar kernel of host operating system. In this virtualization the kernel is able to change dynamically current root file system known as chroot to another root file system without reboot the complete system. The shared kernel virtualization is capable to extend possibly one of main drawback of this virtualization is that the guest operating system must be well-suited with kernel version that is being shared. For example in shared kernel approach it is possible to execute Microsoft windows with this shared kernel approach and it is possible to design Linux guest system for kernel version 2.6 to share a kernel version 2.4. The best example of shared kernel virtualization is Linux Server, Solaris Zones and Containers, FreeVPS and OpenVZ.

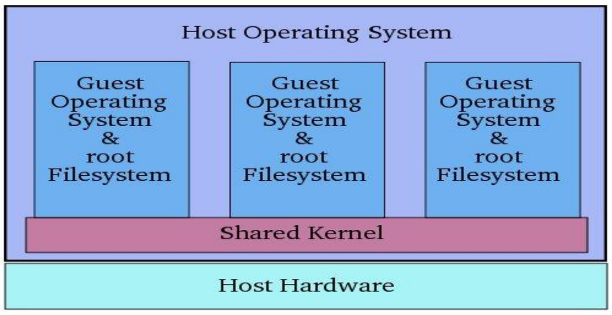

Host Operating System
Guest Operating System & root Filesystem
Guest Operating System & root Filesystem
Guest Operating System & root Filesystem
Shared Kernel
Host Hardware

Fig 2.2: General Diagram of Shared Kernel Virtualization

Kernel Level Virtualization: In this type of virtualization a specific modified kernel execute the host operating system that consists additional designed to manage and control various virtual machines which is containing a guest operating system. Individual guest execute its own kernel while same constraints apply in guest operating system necessary to compile for similar hardware whereas they are running like shared kernel virtualization. In examples it is include Kernel based Virtual Machine (KVM) and User Mode Linux (UML). The below diagram gives a summary of kernel level virtualization

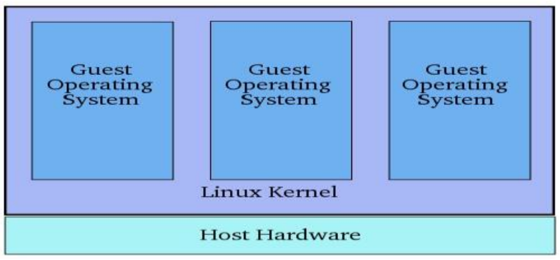


Fig 2.3: General Diagram of Kernel Level Virtualization

Hypervisor Virtualization: The code can run in a range with protection levels which is also referred to as rings provided by CPU family of x86. The operating system kernel typically executes with highest level privilege in Ring 0 and the execution of code in this ring represent to execution in system space, kernel mode and supervisor mode. Rest of the code like execution of application on operating system activates in less privileged ring usually ring 3. The program uses in hypervisor virtualization is hypervisor also referred to as type 1 virtual machine monitor or VMM directly execute on host machine hardware in ring 0. The process used to handle the allocation of memory and resource for virtual machine in extension to provide environment for administration and monitoring tools at higher level. The CPU ring 0 is occupying by hypervisor, the guest operating system uses kernels executing on system necessarily execute in less confidential CPU rings. Maximum kernels of operating system are obviously written to execute in ring 0 for simplest reason which is need to perform tasks those are existing only in ring, like the skill to execute restricted CPU instructions and modified memory directly. There are four types of Hypervisor Virtualization.

Para virtualization: In this type of virtualization guest operating system kernel can be change specially execute on hypervisor. Usually the restricted operation will execute only in ring 0 of CPU with invokes to hypervisor called as hyper calls and hypervisor performs the job on behalf of guest kernel. This naturally restrict to support operating system belongs to open source technology like Linux which modified and registered operating system and owners allowed to make the changes in essential code to target a particular hypervisor. The facility of guest kernel directly communicates with hypervisor results than other virtualization techniques in maximum level of performance.

Full Virtualization: The unmodified guest operating system support are provided by full virtualization and term unmodified represent to operating system kernels that have not change to execute on hypervisor so still run restricted operations as still executing in ring 0 of CPU. In this condition CPU simulation to handle and change restriction also secure CPU actions made by unchanged kernels of guest operating system which is provided by hypervisor. Inopportunely the simulation application needs both resources of time and system to control resultant in substandard levels of performance while compared to that provided by para virtualization.

Hardware Virtualization: This type of virtualization controls properties of virtualization constructed into recent groups of CPUs from both Intel and AMD. These virtualization technologies represent to as Intel VT and AMD-V that provide essential extensions to execute unchanged virtual machine of guest without expenses extend in full virtualization CPU emulation.

In the simplest relationship the new processors provide an extension restrict mode above ring 0 where the hypervisor can operate basically exit ring 0 existing for unchanged guest operating system. Xen, VMware ESX Server and Microsoft's Hyper-V technology are examples of hypervisor based virtualization.

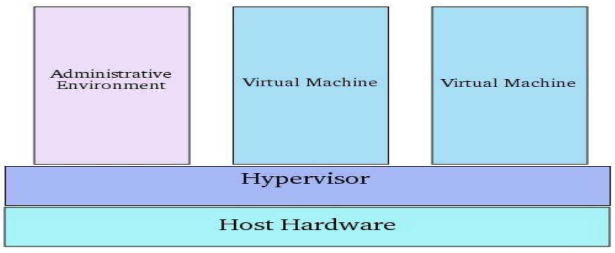


Fig 2.4: General Diagram of Full Virtualization

KVM Hypervisor: It is open source virtualization architecture for Linux distribution and it is represent to virtualization layer in Kernel based virtual machine. A hypervisor is an application which allows various guest operating systems to share hardware of single host. In the virtualization environment of KVM the performance improvement and reorganization management is done by Linux Kernel which is act as Type 2 Hypervisor. The processor, memory, hard disk, network and other resources use coordinates calls and virtual machine environment created by hypervisor through the operating system of host machine. The KVM needs a processor with hardware virtualization additions to connect with guest OS. The KVM has been bundled with Linux operating system can be installed along with Linux Kernel. Various guest operating system can work with kernel virtual machine with Berkeley software distribution, Solaris, Windows, Haiku, ReactOS, Plan 9, and the AROS Research OS. Additionally the QEMU an alternative version can use KVM to execute Mac OSX. It is full virtualization technique for Linux on Intel 64 and AMD 64 hardware which involved kernel of mainline Linux.

The KVM hypervisor supports following characteristics

1. Over-committing: It represents to additional virtualized CPUs and memory allocation in comparison to present system resources.
2. Thin provisioning: It permits to assign the flexible storage and optimizes available flexible storage and optimizes the space for guest virtual machine.
3. Disk I/O throttling: It offers the ability to fix a bound on disk I/O requests sent from virtual machine to host machine.
4. Automatic NUMA balancing: Increase application performance which is running on NUMA hardware systems.
5. Virtual CPU hot add capability: It offers the ability to improve processing power as required on executing virtual machine without interruption.

Resource monitoring setup: The hierarchical design targeted at association of clusters which is represent to ganglia. In a cluster to monitor state depends on a listen/announce protocol based on multicast and uses a tree of point to point between representatives cluster nodes to combine clusters and aggregate their state. A well-known multicast address based for a membership protocol used heartbeat messages by Ganglia within each cluster. The reception of heartbeat is use to maintain membership which represents availability of node and non-reception of heartbeat over a small multiple of periodic announcement interval that is symbol of a node is unavailable. When major modification occurs individual mode monitors its local resources and monitoring data contained by multicast packets to a specific multicast address. Both types of metrics on specific multicast address are listened by all nodes and collection or maintenance for data monitoring of all other nodes. Each node must have approximate view of whole cluster's state and state is simply reconstructed after a crash. Multiple clusters are combined in ganglia together with point to point connections of a tree where each leaf node identified as a node specific cluster being combined whereas nodes higher up in tree specify aggregation points. The copy of individual cluster's monitoring data contained by each cluster node and individual leaf node logically represents a distinct cluster whereas individual non-leaf node logically represents a group of clusters. Multiple cluster nodes to handle failures for each leaf node and aggregation is done by polling child node at specific point in tree at periodic intervals. Data monitoring from both leaf nodes and aggregation points exported using similar process, namely a TCP connection to the node being polled followed by a read of all its monitoring data.

In the implementation process we use mainly two daemons like gmond and gmetad a command line program gmetric and library at client side. The gmond daemon provides single cluster monitoring by executing listen/announce protocol and replying to client requests by returning XML representation of its monitoring data. In a cluster every node executes gmond. On the other hand Ganglia Meta Daemon (gmetad) provides multiple clusters associations. A tree with TCP connections among various gmetad daemons allows monitoring information for various clusters to be aggregated. The application specific metrics can be publish by gmetic command line program used by application whereas client side library provides programmatic access to a subset of Ganglia's attributes.

Components	Values
Processor	Intel(R) Core(TM) i3-5005U CPU@ 2.00GHz
Core	4
RAM	12 GB
OS	UBUNTU 16.04 LTS
OS type	64-bit
Memory	1 TB
Hypervisor	XEN, KVM
Memory(KVM)	100 GB
RAM and CORE(KVM)	4 GB RAM and 2 core
Memory(XEN)	100 GB
RAM and CORE(KVM)	4 GB RAM and 2 core

Table 2.1 EXPERIMENT AND PARAMETERS

PARAMETERS DETAIL

ISSN: 2278-4632
Vol-10 Issue-8 No. 2 August 2020

- **CPU utilization**: Amount of CPU that has been used by each virtual machine on the host 100% represented all CPUs..
  Virtual CPU usage = usage ÷ (number of virtual CPUs × core frequency)
- **Memory Utilization**: For achieving best performance, the host memory was had to be large enough to accommodate the active memory of the virtual machines. The active memory can be smaller than the virtual machine memory in size. That allowed us to over-provision memory, but ensured that the virtual machine active memory was smaller than the host memory.
- **I/O Read**: The number of read input/output operations generated by a process, that included file, network, and device I/O's. I/O Reads directed to CONSOLE (console input object) handles were not counted.
  - **I/O Write:** The number of write input/output operations generated by a process that included file, network, and device I/O's. I/O Writes directed to CONSOLE (console input object) handles were not counted.

3. EXPERIMENTAL RESULTS AND ANALYSIS

Description of scenarios:

This experimentation is analysis of the performances of two hypervisors namely, XEN and KVM on the basis of various parameters used in different load conditions. We have installed Ubuntu 16.04 LTS as host on the system and created virtual machines of Ubuntu 16.04 LTS on both the hypervisors with multiple loads. The scenario for the experimentation is described below:

SCENARIO	PROCESSES
Scenario 1	Ubuntu 16.04 LTS on KVM hypervisor
Scenario 2	Ubuntu 16.04 LTS on XEN hypervisor

Table 3.1 Details of different scenarios

Scenario 1:

In this scenario, we have installed Ubuntu 16.04 LTS on KVM hypervisor as guest with multiple loads. After that results have been evaluated by Ganglia Monitoring tool. The parameters of results are cpu utilization, memory utilization, IO read and IO write time.

Scenario 2:

In this scenario, we have installed Ubuntu 16.04 LTS on XEN hypervisor as guest with multiple loads. After that results have been evaluated by Citrix XEN server. The parameters of results are cpu utilization, memory utilization, IO read and IO write time.

Experimental Details:

For all the results and comparisons of hypervisors to be performed as fairly as possible, some thought has been put into how all the results should be performed. Mainly with regard to each result and the parameters, such as cpu utilization, memory utilization, IO read and IO write time. So project has fix parameter to design the virtual machine after that result evaluated.

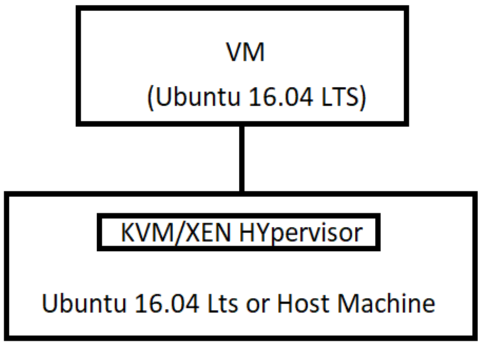


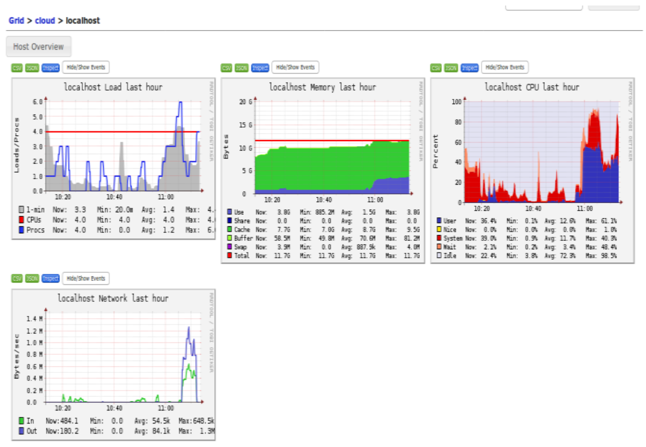
Fig 3.1 Diagram of Experimental Setup

Results and Discussion:

The performance analysis of the scenarios discussed above is as follows:

Scenario 1: Analysis using KVM hypervisor:

In this scenario VM is created on KVM hypervisor and the performance was analyzed using ganglia monitoring tool.



```
1-min  Now:  3.3  Min: 20.0m  Avg: 1.4  Max: 4.
CPUs   Now:  4.0  Min:  4.0   Avg: 4.0  Max: 4.
Procs  Now:  4.0  Min:  0.0   Avg: 1.2  Max: 6.
```

```
Use    Now:  3.8G   Min: 885.2M  Avg:   1.5G   Max:  3.8G
Share  Now:  0.0    Min:   0.0   Avg:   0.0    Max:  0.0
Cache  Now:  7.7G   Min:   7.0G  Avg:   8.7G   Max:  9.5G
Buffer Now: 58.5M   Min:  49.8M  Avg:  70.6M   Max: 81.2M
Swap   Now:  3.9M   Min:   0.0   Avg: 887.9k   Max:  4.0M
Total  Now: 11.7G   Min:  11.7G  Avg:  11.7G   Max: 11.7G
```

```
User   Now: 36.4%  Min: 0.1%  Avg: 12.6%  Max: 61.1%
Nice   Now:  0.0%  Min: 0.0%  Avg:  0.0%  Max:  1.0%
System Now: 39.0%  Min: 0.9%  Avg: 11.7%  Max: 40.3%
Wait   Now:  2.1%  Min: 0.2%  Avg:  3.4%  Max: 48.4%
Idle   Now: 22.4%  Min: 3.8%  Avg: 72.3%  Max: 98.5%
```

```
In   Now:484.1  Min:  0.0  Avg: 54.5k  Max:648.5k
Out  Now:180.2  Min:  0.0  Avg: 84.1k  Max:  1.3M
```

Fig. 3.2: Local host report of load, memory, C.P.U. and network.

D.O.I : 10.46528/JK.2020.V10I08N02.08

ISSN: 2278-4632
Vol-10 Issue-8 No. 2 August 2020

Table 3.2 Ganglia local host report

NODE	MEMORY	VALUES	CPU	VALUES
LOCALHOST	USE	1.5G	USER	12.6%
	SHARE	0.0	NICE	0.0%
	CACHE	8.5G	SYSTEM	11.7%
	BUFFER	70.6M	WAIT	3.4%
	SWAP	887.9K	IDLE	72.3%

Summary of figure 3.2: This figure shows the details of the localhost in terms of load, memory, CPU and network for the last one hour. In this observation the load is represented in load per processes, the memory used in bytes, the CPU utilization in percentage and network load in bytes/sec.

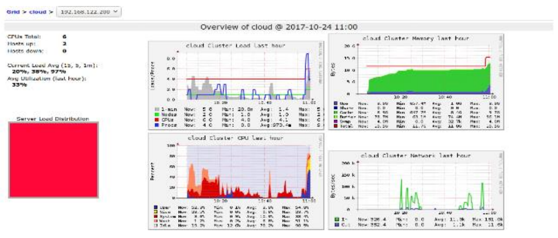

Overview of cloud @ 2017-10-24 11:00
cloud Cluster Load last hour
cloud Cluster Memory last hour
cloud Cluster CPU last hour
cloud Cluster Network last hour
Server Load Distribution

Fig 3.3: Network Metrics

Summary of figure 3.3: This figure shows the network metrics and giving the details about bytes received, bytes sent, packets received, packets sent, total running processes and total processes.

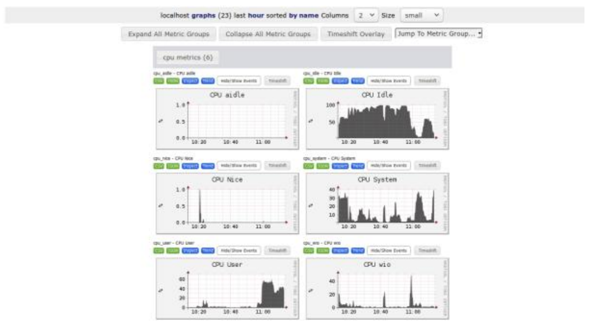


Fig 3.4: CPU Metrics

Summary of figure 3.4: This figure shows the CPU metrics and giving details about CPU an idle time, CPU idle time, CPU nice time, CPU system usage, CPU user time and Input/output values.

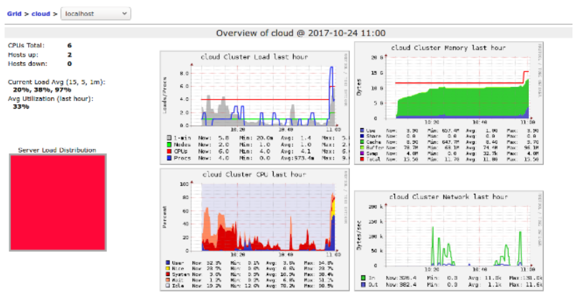

Overview of cloud @ 2017-10-24 11:00
cloud Cluster Load last hour
cloud Cluster Memory last hour
cloud Cluster CPU last hour
cloud Cluster Network last hour
Server Load Distribution

Fig 3.5: Cluster overview of localhost

Summary of figure 3.5: This figure shows the cloud cluster load, cloud cluster memory performance graph, cloud cluster network and cloud cluster CPU performance graphs for the localhost.

Fig 3.6: Cluster overview of guest (192.168.122.200)

NODE	MEMORY	VALUES	CPU	VALUES
LOCALHOST	USE	657.4M	USER	3.6%
	SHARE	0.0	NICE	0.6%
	CACHE	647.7M	SYSTEM	10.9%
	BUFFER	63.1M	WAIT	6.8%
	SWAP	0.0	IDLE	78.2%

Table 3.3 Ganglia local host report 2

Summary of figure 3.6: This figure shows the cloud cluster load, cloud cluster memory performance graph, Cloud cluster network and cloud cluster CPU performance graphs for the guest (192.168.122.200). Here the total CPUs are 6, Hosts up are 2, current load average are 20%, 38% and 97% and average utilization is 33%.

D.O.I : 10.46528/JK.2020.V10I08N02.08

ISSN: 2278-4632
Vol-10 Issue-8 No. 2 August 2020

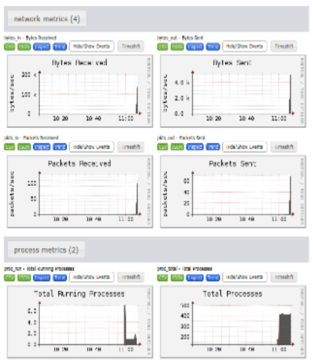


Fig 3.7: Network Metrics

Summary of figure 3.7: This figure shows the network metrics and giving the details about bytes received, bytes sent, packets received, packets sent, total running processes and total processes for the guest.

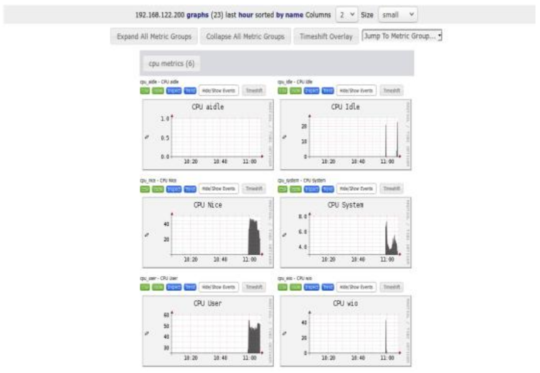


Fig 3.8: CPU Metrics

Summary of figure 3.8: This figure shows the CPU metrics and giving details about CPU aidle time, CPU idle time, CPU nice time, CPU system usage, CPU user time and Input/output values for guest.

5.3.2 Scenario 2: Analysis using XEN hypervisor:

The results obtained during the performance analysis on citrixXenserver are given below:

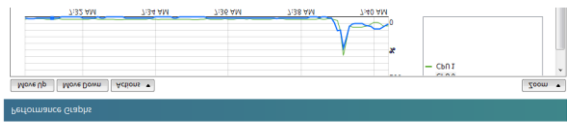


Fig 3.9: Performance graph

Summary of figure 3.9: This figure shows the performance graph for the virtual machine running on Xen server.

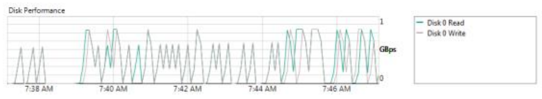


Fig 3.10: Disk performance graph (in GBps)

Summary of figure 3.10: This figure shows the disk read/write performance(in GBps) for the VM(s) installed on Xen server.

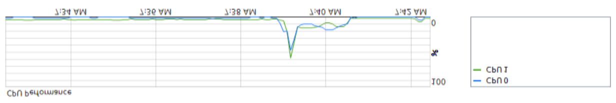


Fig 3.11: CPU performance graph (%)

Summary of figure 3.11: This figure shows the CPU performance of the Xen server VM(s) on increased load.

Comparison of the performances of KVM and XEN on the basis of parameters:

PARAMETERS	KVM	XEN
C.P.U	**0.999ms**	**0.993ms**
MEMORY UTILIZATION	**10 GB**	**8.5 GB**
IO WRITE	**0.855GBps**	**0.834GBps**
IO READ	**0.852GBps**	**0.894GBps**

Table: 3.4 Comparisons between KVM and XEN

4. RESULTS:

In this work, we have installed Ubuntu on two hypervisors i.e. KVM and XEN. There we have compared the performance of both the hypervisors on the basis of following parameters i.e. Disk I/O, CPU performance, Network performance and memory management. Firstly, we have installed ganglia in our host system to measure the performance of the guest and the host machine. Then we have installed one VM and measured the performance and then we installed another VM to increase the load and checked the performance of the hypervisor. Secondly, we have installed Citrix server and then installed one virtual machine and checked the performance and then installed another VM and measured the performance on the basis of parameters. In the results, we have presented a quantitative comparison of Xen and KVM focused on overall performance, performance isolation, and scalability. The most striking difference that we have found between the two systems was in scalability. KVM had substantial problems with guests crashing, beginning with 4 guests. KVM had better performance isolation than Xen, but Xen's isolation properties were also quite good. The overall performance results were mixed, with Xen out- performing KVM on a kernel compile test and KVM out- performing Xen on I/O-intensive tests. We would like to extend our comparison to include Xen with full virtualization (HVM) and KVM with para virtualized I/O.

D.O.I : 10.46528/JK.2020.V10I08N02.08

REFRENCES

[1] Fredric Stumpf and Claudia Eckert, “Enhancing Trusted Platform Modules with Hardware-Based Virtualization Techniques”, The Second International Conference on Emerging Security Information, Systems and Technologies, IEEE 2008.

[2] Irfan Ahmad and Palo Alto,” Easy and Efficient Disk I/O Workload Characterization in VMware ESX Server”, IEEE Conference, 2007

[3] Ao Ma, Yang Yin, Wenwu Na, Xiaoxuan Meng, Qingzhong Bu1 and Lu Xu1, “Scrubbing in Storage Virtualization Platform for Long-term Backup Application”, International Conference on Availability, Reliability and Security, 2009.

[4] Sébastien Varrette, Mateusz Guzek, Valentin Plugaru, Xavier Besseron and Pascal Bouvry,” HPC Performance and Energy-Efficiency of Xen, KVM and VMware Hypervisors”, 25th International Symposium on Computer Architecture and High Performance Computing, 2013.

[5] Thomas Muller and Alois Knoll,” Virtualization Techniques for Cross Platform Automated Software Builds, Tests and Deployment”, Fourth International Conference on Software Engineering Advances, 2009.

[6] Karim Chine,”Scientific computing environments in the age of virtualization toward a universal platform for the cloud”, OSSC, 2009.

[7] J. Brandt, F. Chen, V. De Sapio, A. Gentile, J. Mayo, P. P´ebay, D. Roe, D. Thompson, and M.Wong,” Combining Virtualization, Resource Characterization, and Resource Management to Enable Efficient High Performance Computer Platforms Through Intelligent Dynamic Resource Allocation” IEEE 2010.

[8] Ziye Yang, Haifeng Fang, Yingjun Wu, Chunqi Li, Bin Zhao” Understanding the Effects of Hypervisor I/O Scheduling for Virtual Machine Performance Interference” , IEEE 4th International Conference on Cloud Computing Technology and Science, 2012.

[9] Sébastien Varrette, Mateusz Guzek, Valentin Plugaru, Xavier Besseron and Pascal Bouvry“HPC Performance and Energy-Efficiency of Xen, KVM and VMware Hypervisors” 25th International Symposium on Computer Architecture and High Performance Computing, 2013.

[10] Alexandru-Cătălin Bujor and Răzvan Dobre “KVM IO Profiling” IEEE 4th International Conference on Cloud Computing Technology and Science, 2012.

[11] Gauri Joshi, S.T. Shingade and M.R. Shirole ” Empirical Study of Virtual Disks Performance with KVM on DAS IEEE International Conference on Advances in Engineering & Technology Research ICAETR, 2014.

[12] Jinho Hwang, K. K. Ramakrishnan and Timothy Wood," NetVM: High Performance and Flexible Networking Using Virtualization on Commodity Platforms", IEEE transactions on network and service management, vol. 12, no. 1, march 2015.

[13] P. Vijaya Vardhan Reddy, Lakshmi Rajamani “Performance Comparison of Different Operating Systems in the Private Cloud with KVM Hypervisor Using SIGAR Framework” International Conference on Communication, Information & Computing Technology (ICCICT), Jan. 16-17, Mumbai, India.

[14] Michele Stecca, Corrado Moiso, Martino Fornasa, Pierpaolo Baglietto, and Massimo Maresca,” A Platform for Smart Object Virtualization and Composition” IEEE Internet of things Vol. 2 No. 6, December 2015.

[15] Moritz Raho, Alexander Spyridakis, Michele Paolino, Daniel Raho,” KVM, Xen and Docker: a performance analysis for ARM based NFV and Cloud computing” 978-1-5090-1201-5/15/$31.00 c
2015 IEEE.