Analysis of Performance in the Virtual Machines Environment

Praveen G, Prof. Vijayrajan
School of Computing Science and Engineering
VIT University,Vellore, Tamil Nadu, India
Praveen1600@gmail.com, vijayarajan.v@vit.ac.in

Abstract

In this paper, we propose a scheme that manages the Performance and resource of virtual machines that are used to host computing applications. As the size and complexity of modern computing systems keep increasing to meet the demanding requirements of Performance applications, manageability is becoming a important concern to achieve both performance and productivity computing. Meanwhile, virtual machine (VM) technologies have become well-liked in both industry and academia due to various features designed to ease system management and administration. While a VM-based environment can greatly help productivity on large-scale application system. In this paper, we provide an analytic framework for the performance analyzing either without running a system or in a run able real system. With these market trends toward virtual environments, many research groups are developing evaluation tools to check the performance of virtual systems and their overheads. However, the performance characterization in virtual environments has not been established yet for many challenging issues. However, achieving close to peak performance requires careful attention to a plethora of system details. The benefits of virtualization are typically considered to be server consolidation, (leading to the reduction of power and cooling costs) increased availability, isolation, ease of operating system deployment and simplified disaster recovery. One main challenge for virtualization, the need to maximize throughput with minimal loss of CPU and I/O efficiency.

Keywords :*Virtual machine monitor, XEN Architecture, CPU Utilizations, Virtual memory, Virtualization.*

1. Introduction

In computer science, virtual machines recover the interests to researches as well as the industries. The main advantages of the virtual machines are that multiple guest operating systems can co-exist on the same computer and it is in strong isolation from each other. Another important benefit of the virtual machines is to provide an instruction set architecture (ISA) that could be different from that of the real machine. Applications of virtual machines arise widely in computer science, such as in embedded systems, which support a real-time operating system at the same time as a high-level OS. In virtual machines(VMs) environment, the software layer which provides the virtualization is called a *virtual machine monitor* (VMM).The guest OS do not have to be all the same, they could be different OS or different version of an OS on the same computer. These nature or type of virtual machines are very useful for the server consolidations. To the best of our knowledge, the main types of virtualization technologies are full-virtualization , Para-virtualization , pre-virtualization, hardware virtualization. In full virtualization, the virtual hardware is affinity to the authentic physical hardware. This allows the guest OS do not need to be modified. VMware uses this technology. In Para-virtualization, the guest OS is modified to evasive some performance overhead. In pre-virtualization which combines the performance of Para-virtualization with the modularity of backward virtualization. In hardware-virtualization, Intel and AMD have independently developed virtualization extensions to the x86 architecture. No matter what technologies we consider for the virtual machines, one task is to quantify and predict the performance of the system. The knowledge of ability is to monitor the resource usage of different VMs. However, designing an analytic performance model to predict the performance is of great interests to the designer.

With the modularity of virtualization technology, VM (*Virtual Machine*) has the following properties, such as flexible, secure, reliable, and easy to manage and configure, which can greatly minimized the hardware cost. However, besides these benefits, it also increases the complexity of virtual computing systems, and maximizes the performance of virtual computing systems. Furthermore, end user desktop will also receive the benefits of virtualization. For desktop users, the output of high-performance virtual machine display is an important metrics[3] for application. Nevertheless, there is a difference between the virtual environment of the display mechanism and the traditional computer display mechanism, which impact into the fact that we cannot copy the evaluation methods and techniques of the traditional computer. An important problem as how to effectively and evaluate virtual machine display performance pops up. If you expected that virtualization will come flawless, then you didn't have realistic expectations. Virtualization is great and it can do a lot of good for your data center but in order to take advantage of its benefits, you need to know how to deal with some of the problems that come with it.

Virtualization is a relatively young technology and even though it is adopted in many enterprises, there are still lots of technical challenges that need to be overcome. Discovering the reason for performance degradation is not always easy and this alone can cause a lot of problem in a production environment. Both server and storage virtualization come with issues but their nature differs[6].

Performance must be tuned for different types of application workloads[4]. Tuning for large files and continuous performance isn't optimal for small files and vice versa, and tuning for high performance of small files isn't best for large files and continuous performance. If one or two VMs are running especially I/O-intensive applications and hammering a disk, other VMs that utilize the same data store may suffer the effects of disk I/O contention. This problem is more difficult to identify than overloaded CPU or memory. Increase the memory size and processor cores (if supports) to guest operating system. You have to consider host usage while allocating memory, but processor cores can be set to maximum without considering host usage. Nowadays virtualization is ubiquitous and virtualization technologies play an important and significant role in many IT fields. The main advantages of virtualization is it can rapidly reduce cost and uncertain of the experiments, portability of a virtual machine to another is simple, it has improved security, it enables parallelization, it decreases time expenses needed for administration of a large amount of desktops and workstations, etc. The virtual machine is a technology that creates one or multiple virtual environments on a single physical machine[5]. The virtual machines are separated from each other and the underlying physical machine, and they give users the illusion of accessing a real machine directly.

2. Virtual Machine Monitor

A virtual machine monitor (VMM) is a host program that allows a single computer to support multiple, identical execution environments. All the users see their systems as selfcontained computers isolated from other users, even though every user is served by the same machine. In this context, a virtual machine is an operating system (OS) that is managed by an underlying control program. In Microsoft Virtual Server 2005, Virtual Machine Monitor is the proprietary name for a kernel-mode driver that functions as a firewall between the host OS and the virtual machines. It can avoid any single program, running in one of the virtual machines, from overusing the resources of the host OS.

The software that creates a virtual machine environment in a computer. In a regular, non-virtual environment, the operating system is the master control program, which manages the execution of all applications and acts as an interface between the applications and the hardware. In a virtual machine environment, the virtual machine monitor (VMM)[10] becomes the master control program with the maximum privilege level, and the VMM manages one or more operating systems, now referred to as "guest operating systems." Each guest OS manages its own applications as it normally does in a non-virtual environment, except that it has been separated in the computer by the VMM. Each guest OS with its applications is known as a "virtual machine" and is sometimes called a "guest OS stack."

The VMM flexibility and performance in a virtual machine environment, because it is not considered to any limitations of a host OS. It relies on its own software drivers for the hardware for ultimate control. however, they may limit portability to another hardware platform if the hypervisor does not support the peripherals on that platform.

In hardware virtualization, the concept host machine refers to the actual machine on which the virtualization takes place. The term guest machine, however, refers to the virtual machine. Likewise, the adjectives host and guest are used to help distinguish the software that run on the actual machine from those that run on the virtual machine. The software that creates a virtual machine on the host hardware is called Hypervisor[1] or Virtual Machine Monitor.

3. Analytic Framework

In this section we give a framework to analyze the performance of general virtual machines environment. The analytic queue network results are sometimes criticized for inaccurate performance evaluation and then it is invaluable during the development process[7]. However, without analytic results, it is too cost for the designers to test every new design proposal using all kinds of experiments. Furthermore, the designer cannot get the performance guarantees for any new design. Consequently, it is important for the designer to yield some analytic results though there is a gap between practice and theory. The virtualized system consists total *n* virtual machines above the VMM (or hypervisor). Each virtual machine has its own operating system. The detail of the framework is illustrated in Fig. 1. In the following, we will adopt the queue network theory[2] to analyze the performance of the given framework. The framework consists of four components, input, workload analysis or monitor, service and performance analysis. In the input part, we must specify the queue network model. Then the workload in each virtual machines are described. In the case of single class, the workload is described as the work intensity[4] for each resource. In the case of multiple classes, a matrix of work intensive for every resources are required.

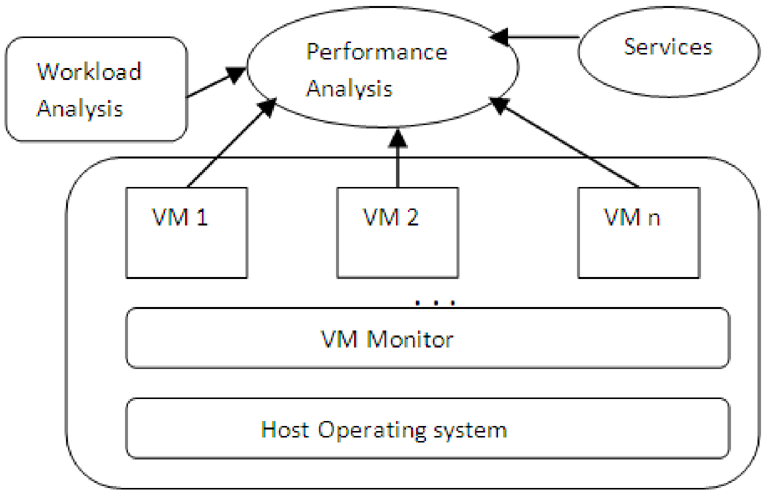

Figure 1: Analytic Framework

In the service parts, the service of the input workload are requested[11], i.e. the service time of a class. Note that, in this part, the service of physical hardware devices for each class of workload are request to be known.In the workload analysis or monitor part, one of them is selected for the final part performance analysis. In the workload analysis part, we suppose that the final output is a sequence of instructions. In the monitor part, we suppose to run a monitoring program to record down the utilizations of virtual devices in all the virtual machines. Unlike the workload analysis part, the virtual system in the monitor part must be runnable. Finally, all the information from the first three parts pass to the component performance analysis. The performance metrics on each class and each device could be utilization, throughput, queue length, response time. We also concern on aggregate system's performance of each virtual machine.

4. Performance Analysis

In the general virtualization environment, each virtual machine runs several classes of jobs, our task is to estimate the performance in each virtual machine. Though jobs running in each virtual machine don't know whether the other virtual machines is running or not, but in the analyze model. On the other hand, we assume that the hypervisor[1] adopts the first come first server (FCFS) scheduling policy to assign the incoming jobs for multiple classes model. A case study has been conducted on the Xen Architecture to analyze the performance of the VMM.

5. Xen Architecture

Xen is a free and open-source virtual machine monitor (VMM) which allows multiple (guest) operating system (OS) instances to run concurrently on a single physical machine[5]. It uses Para virtualization, where the VMM exposes a virtual machine abstraction slightly different from the underlying hardware. The Xen system has multiple layers. Each guest OS runs on a separate virtual machine called domain. Each application, running on a guest OS, accesses hardware devices via a special privileged virtual machine called isolated driver domain (IDD), which owns specific hardware devices and run their I/O device drivers. All other domains (guest domains in Xen terminology) run a simple device driver which communicates with the driver domain to access the real hardware devices.

The IDD can access directly the hard-ware devices it owns. However, a guest domain accesses the hardware devices indirectly through a virtual device connected to the IDD.[10]

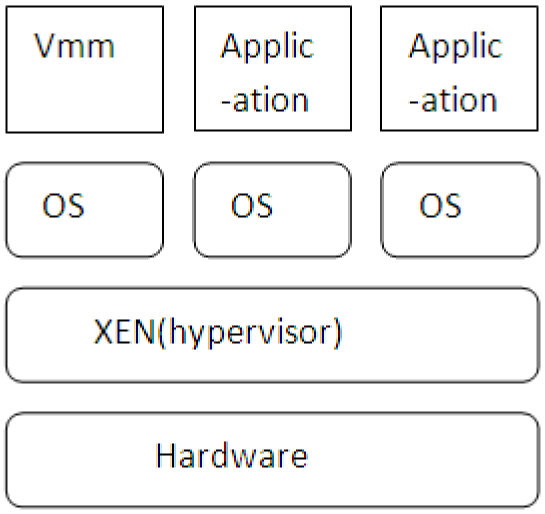

Figure 2: Xen Architecture

A case study has been conducted and reviewed [7]. In order to demonstrate the causes of virtualization overhead on the Xen VMM, in the VM CPU utilization is slightly higher than the sum of the utilizations of the two CPUs . Note that there is a significant load on the IDD CPU, since it works as interface for the hardware to the VM, which is also the sum of the CPU utilizations . Since this does not execute a representative number of I/O operations, the CPU utilization on the IDD is almost 0. Note that, but there is a considerable CPU processing on the IDD due to network I/O operations.

Based on these observations, we can conclude that the assignment of CPU resources to VM and IDDs can affect critically system performance, since the IDD processing is significant for workloads which stress I/O operations.

In the virtual environment, we assign one CPU for the IDD and one CPU for the guest VM. Note that CPU utilization increases linearly with request rate for both IDD and VM. As we will further discuss, the ratio between the CPU demands (and thus CPU utilizations) for the IDD and the VM is constant over the range of request rates. As one might expect, The average response time is significantly longer at the virtual environment for request rates greater than 7000 requests per second[7], when the server starts to become overloaded. Clearly, the VM is the bottleneck in the Xen system, and performance degrades significantly with 7800 requests/sec. The Xen virtual environment is able to provide the same throughput at the cost of a higher CPU utilization[9].

The Performance can be benchmarked even by considering the Idle Response, Idle Response since requested and Idle time metrics. When the Idle Response is low, cpu utilizations is considerably low and thus provides the information for the system. If the metric[3] such as cpu is low then, the size of the virtual memory also decreases. Note that it is not true in all the cases when the cpu is maximized but, the virtual memory can be low or viceversa. It depends on the amount of the work analyzed by the server and the IDD.

CPU hardware and features are rapidly increasing, and your performance testing and analysis methodologies may need to evolve as well. CPU utilization is a key performance metric[3]. It can be used to track CPU performance improvements, and is a useful data point for performance problem investigations. CPU utilization has critical implications on other system performance characteristics, namely power consumption. We may think the magnitude of CPU utilization[3] is only important if you're bottlenecked on CPU at 100%, but that is not at all the case. Each additional % of CPU Utilization consumes a bit more[8]. In order to demonstrate the causes of CPU Utilization , in the Fig3(a) a graph has been analyzed with the CPU Utilizations within the threshold values. From the information with the graph we can infer that the maximum CPU has been utilized is up to 12% and the average CPU utilization is 5 % . It allows other process and task to run with more productivity and efficiency as the given process has not been consumed[3] excess of the cpu utilizations. In the XEN environment, the VM CPU utilization is slightly higher than the sum of the utilizations of the two CPUs[7] . Note that there is a significant load on the IDD CPU, since it works as interface for the hardware to the VM, which is also the sum of the CPU utilizations. In the Fig3(b), a graph has been analyzed with the CPU Utilizations beyond the threshold values. From the observation with the graphs, the CPU utilization has reached a maximum value of 100% utilizations and a Average utilization of 35%.The VM CPU utilization is higher. That makes the other process slow for their IO operations. In the VM, allocation of work to cores is done by the hypervisor rather than the guest OS[7]. If you want to view CPU utilization information via Performance Monitor, specific hypervisor-aware performance counters should be used. In the VM, unique Perfmon counters exposed by the hypervisor to the root partition should be used to get accurate CPU utilization information. Perfmon is a good starting point for measuring utilization but it has several limitations that can make it less than optimal[2].

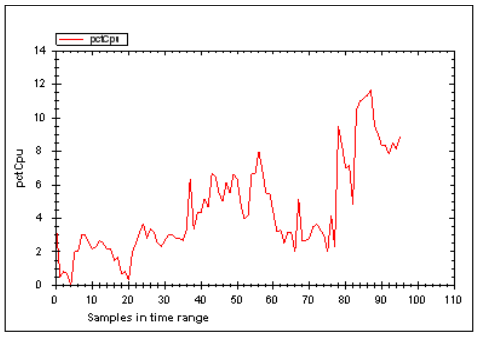

Figure 3(a): Graphs for CPU Utilizations with Threshold Values

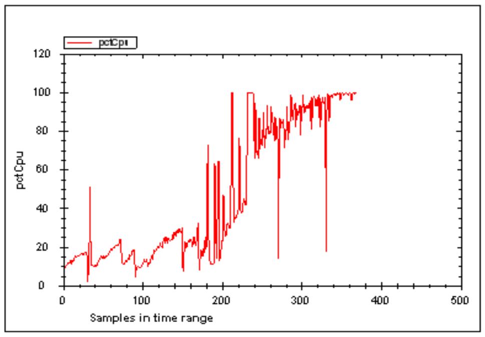

Figure 3(b): Graphs for CPU Utilizations Beyond the Threshold Values

In the Idle time, i.e. the amount of time the CPU is not utilizing the cpu cycles[6]. In the performance of the vm environment, consideration of idle time certainly may be useful when we consider the virtual memory. In the fig 6,a graph has been analyzed with respect to Idle time and the samples in time range. When there is no load or minimum load(which is negligible) we observe that some amount of system has been utilized. The system is not Idle completely (which is not possible).Based on the observation that, the system is utilizing some amount of time for some time interval when there is no consideration of load. when the load is increased gradually, the Utilization attains its peak value for the respective time intervals[9].

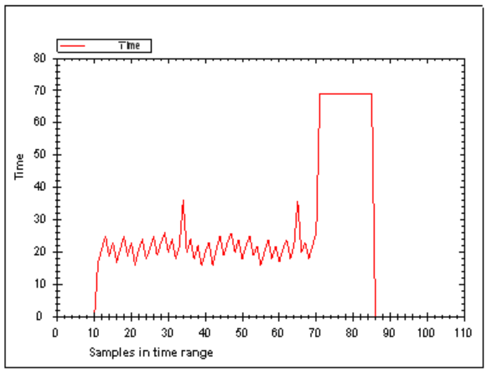

Figure 6:Graphs for Idle Time Utilizations

In the performance analyzes of the vm, threads that are scheduled by a virtual machine instead of natively by the underlying operating system. Threads emulate multithreaded environments without relying on any native OS capabilities, and they are manageable in user space instead of kernel space ,enabling them to work in environments that do not have native thread support.

Performance On a multi-core processor, native thread implementations can automatically assign work to multiple processors, threads can be started much faster on some VM.

Also, a thread may block all other threads if performing a blocking I/O operation. To prevent the problem, threads must use asynchronous I/O operations, although the increased complexity can be hidden by implementing separate native I/O processes which cooperate with threads.

In the fig 4(a),a graph has been observed that the number of threads within the threshold values is limited with the performance of the vm. The Xen virtual environment is able to provide the same performance at the cost of a higher time samples. In the fig 4(b),a graph has been observed that ,the number of threads beyond the threshold values rises its utilization for a short period of time and attains the constant value for the remaining time samples.

Performance test is analyzing the load characteristics of the system under test . In this, the main challenge is to generate the expected load without having as complex hardware as the system is itself. It usually have the assumption that the load can be generated by deploying more hardware resources, which improves the efficiency of scheduling in this performance test environment.

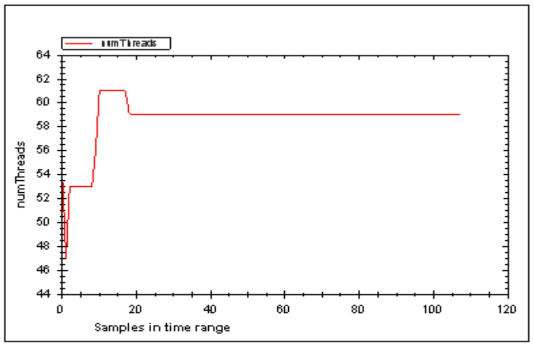

Figure 4(a): Graph for Number of Threads Beyond Threshold Values.

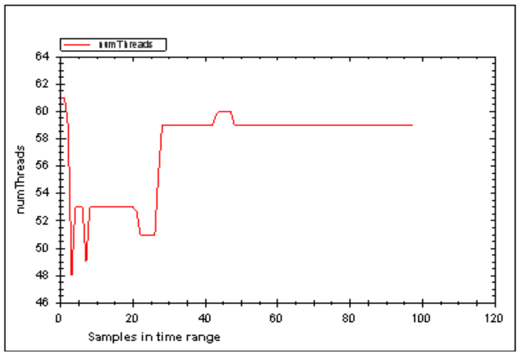

Figure 4(b): Graph for Number of Threads Within the Threshold Values.

In Performance analyzes, input/output, or I/O, refers to the communication between an information processing system and the outside world, possibly a human, or another information processing system. Inputs are the signals or data received by the system, and outputs are the signals or data sent from it. The term can also be used as part of an action to "perform I/O" is to perform an input or output operation.

Note that the designation of a device as either input or output depends on the perspective. Mouse and keyboards take as input physical movement that the human user outputs and convert it into signals that a computer can understand. The output from these devices is input for the computer.

During the operation of a data processing system capable of multi-tasking, a count is made of the number of times each I/O device is accessed by each task. The counting is done over the time interval between successive allocation routines. During each allocation, an analysis is made using the count and time interval to estimate the utilization of each device due to the current tasks. An estimate is also made of the anticipated utilization due to the task undergoing allocation. The estimated current and anticipated utilization are then considered and an attempt is made to allocate data sets to the least utilized I/O devices so as to achieve balanced I/O activity.

To compute the accuracy of I/O associations within the VMM. These traces were compared to corresponding traces of inferred VMM associations to calculate the accuracy of the mechanisms. To stress the I/O association, ability of the VMM, we increase the load on the system and plot the resulting I/O association accuracy[11]. Figure 5, shows our results. For low levels of concurrency, the simple context association method achieves a high degree of accuracy. With a large number of process groups, however, the accuracy of the context method reduces dramatically increased queuing delays between an I/O being issued and its observation by the VMM cause a majority of the requests to be incorrectly associated with other (CPU-bound) processes. Event chaining, on the other hand, is able to achieve nearly perfect accuracy regardless of the level of concurrency.

The objective of controlling I/O subsystem operation suggests a need for selecting a variable that measures what it is to be controlled, I/O load and I/O utilization. Each of these variables measures I/O subsystem operation and, as a result are directly related so that a change in any one variable will signify a reasonably proportionate change in the other two. The I/O load is perhaps the variable most commonly associated with controlling I/O subsystem operation and it refers to the rate of demand for an I/O resource. However, the I/O load is extremely complex[11], when I/O requests are queued up and are then not handled on a first in, first out basis so that consideration should be made of such factors as queue lengths, intervals between arrival times, the time in the queue and service time. It is more profitable to concentrate on the effect that load is having on the system's resources rather than on the load itself. Thus, the variable "I/O resource utilization," which measures the effect caused by load, was selected as the variable that provides a measure of what is to be controlled. By improving the performance by achieving the balanced I/O utilization[11].

In the fig 5, a graph has been analyzed with the observation that the load on the I/O Utilization is excess and thus it provides the imbalanced I/O utilizations. The I/O utilizations have been attained its maximum value during the peak load in the system ,thus the VM is able to withstand the excess load and the graph result summarizes the I/O utilizations.

The objective of the measurement and analysis processes are to extrapolate what the I/O utilization will be sometime in the near future. Obviously, one end point of the time interval should be as close to the point in time when the measurement will be used. This means that the one end of the measurement interval will occur during the execution of the I/O device allocation routine. However, the problem remains as to what the starting point should be of this time interval. If the time interval is too short, there is a chance that some utilization patterns will be excluded. On the other hand, if the time interval is too long, there is a lesser possibility that, due to chance or normal irregularities in the process, some significant source of utilization may either go unnoticed.

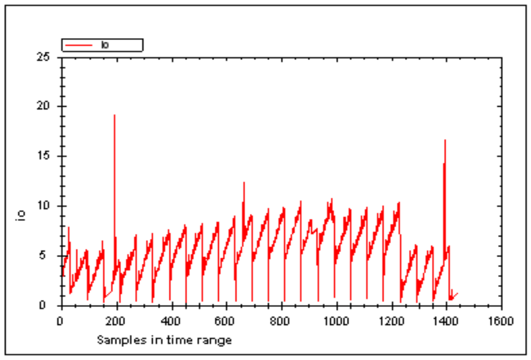

Figure 5: Graphs for I/O Utilizations with Excess Load

In the performance analyses of the VM, Virtual memory plays a important significant role. virtual memory is a memory management technique developed for multitasking kernels. This technique virtualizes a computer architecture's various hardware memory devices (such as RAM modules and disk storage drives). Virtual memory is an integral part of a computer architecture all implementations (excluding emulators and virtual machines) require hardware support, typically in the form of a memory management unit built into the CPU. Consequently, older operating systems generally have no virtual memory functionality[12].

Virtual memory systems trigger unpredictable interrupts that may yield unwanted "jitter" during I/O operations. This is because embedded hardware costs are often kept low by implementing all such operations with software rather than with dedicated hardware. In any case, embedded systems usually have little use for complicated memory hierarchies. Virtual memory enables each process to act as if, it has the whole memory space to itself, since the addresses that it uses to reference memory are translated by the virtual memory mechanism into different addresses in physical memory. The size of the virtual memory on a system is smaller than the sum of the physical RAM

Virtualization of OS images will become more and more prevalent, this means another layer of memory handling is added to the picture. Virtualization of processor OS containers do not fall into this category since only one OS is involved. Technologies like Xen enable with or without help from the processor . In these situations there is one piece of software alone which directly controls access to the physical memory.

In the case of Xen, the Xen VMM is that piece of software. The VMM does not implement many of the other hardware controls itself . Unlike VMM on other, earlier systems (and the first release of the Xen VMM) the hardware outside of memory and processors is controlled by the privileged domain.

Based on the observations, in the Fig 8(a),when the load is with in the threshold value the virtual memory initiates with the 2.05 GB and it gradually increases to 2.25 GB for a short span of time and the virtual memory decreases as the load is with in the threshold value. When the load is stabilized with the threshold value ,the graph is constant with its respective load with 2.28 GB as observed by the graph.

To implement the separation of the domains which is required for the virtualization to be complete. The VMM does not hand out memory by giving out individual physical pages and letting the guest OS handle the addressing this would not provide any protection against faulty or rogue guest domains. Instead, the VMM creates its own page table tree for each guest domain and hands out memory using these data structures. The good thing is that access to the administrative information of the page table tree can be controlled. If the code does not have appropriate privileges it cannot do anything.

In the fig 8(b),A graph has been analyzed with the load beyond its threshold value. When the load is increased the virtual memory rises to the value of 2.6 GB and stabilizes with 2.61 GB for certain span of time. As the load is increased beyond the threshold value, the potential of the virtual memory goes to its peak value 2.63 GB and retains more utilization of the virtual memory for some amount of time until the load is decreases and attains its minimum value and gets stabilized with the load.

This access control is exploited in the virtualization Xen provides, regardless of whether Para or full virtualization is used. The guest domains construct their page table trees for each process in a way which is intentionally fairly similar for Para-virtualization. Whenever the guest OS modifies its page tables the VMM is invoked. The VMM then uses the updated information in the guest domain to update its own shadow page tables. These are the page tables which are actually used by the hardware. Obviously, this process is quite expensive, each modification of the page table tree requires an invocation of the VMM. While changes to the memory mapping are not cheap without virtualization they become even more expensive now.

The additional costs can be really large, considering that the changes from the guest OS to the VMM and back themselves are already quite expensive physical addresses. This will allow memory handling at almost the speed of the no-virtualization case since most VMM entries for memory handling are removed. It also reduces the memory use of the VMM since now only one page table tree for each domain (as opposed to process) has to be maintained.

The amount of work needed for each address space modification is one problem with virtualized OS. There is another problem inherent in VMM-based virtualization, though, there is no way around having two layers of memory handling. The Xen approach of using a separate VMM makes optimal (or even good) handling hard since all the complications of a memory management implementation, including "trivial" things like discovery of memory regions, must be duplicated in the VMM.

There is no separate VMM running directly on the hardware and controlling all the guests, instead, a normal kernel takes over this functionality. This means the complete and sophisticated memory handling functionality in the kernel is used to manage the memory of the system. Guest domains run along side the normal user-level processes in what the creators call "guest mode". The virtualization functionality, Para or full virtualization, is controlled by another user-level process.The benefit of this model over the separate VMM of the Xen model is that, even though there are still two memory handlers at work when guest OS are used, there only needs to be one implementation, that in the kernel. It is not necessary to duplicate the same functionality in another piece of code like the Xen VMMOverall, programmers must be aware that, with virtualization used, the cost of memory operations is even higher than without virtualization. Any optimization which reduces this work will pay off even more in virtualized environments.

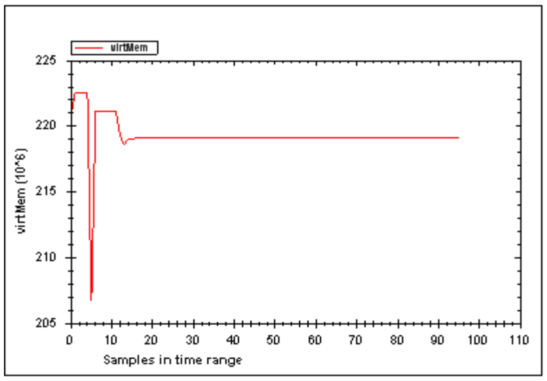

Figure 8(a): Graphs for the Virtual Memory with the Threshold Value

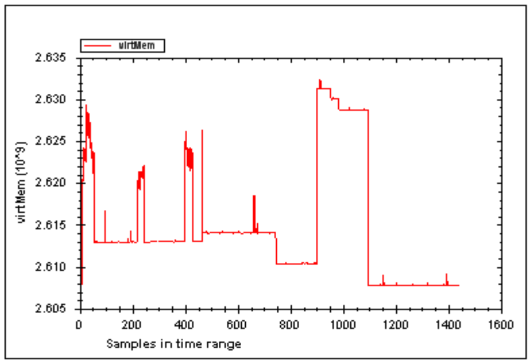

Figure 8(b): Graphs for the Virtual Memory Beyond the Threshold Value

6. Conclusions and Future Work

This work proposes and validates simple analytic performance to predict how applications will perform on Xen VMs, based on the performance of applications running on non-virtual environments. We envision two directions towards which our work can evolve. In other words, our models can be generalized to represent the overhead of the IDD performed on behalf of application programs as a special workload class the service demands of the special overhead class are load dependent. Second, we believe that our models can support the design of performance predictor tools as well as self-adaptive virtual systems[7]. The main idea is to obtain metrics from the non-virtual environment and based on previously established information, such as slowdown, to estimate the performance of the Idle time utilizations, virtual memory[12],CPU utilizations. Though our study is based on Xen virtual machine monitor, the approach is general enough to be applicable to other virtualization technologies as well, the study has made benefits with more prominently visible in the VM's CPU usage .The analysis has been estimate for the metrics with CPU usage, virtual memory, Idle time with respect to performance. The future work includes further tuning on our system to make it accommodate the applications that run concurrently, and making further study on the co-exist situation for multiple virtual machines computing applications.

References

[1] Younggyun Koh, Rob Knauerhase, Paul Brett, Mic Bowman, Zhihua Wen, Calton "An Analysis of Performance Interference Effects in Virtual Environments"

[2] Deshi Ye, Q. He, Hua Chen, Jianhua Che "A framework to evaluate and predict performances in virtual machines Environment". 2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing

[3] Zhiyuan Shao, Hai Jin, Xiaowen Lu "PMonitor: A Lightweight Performance Monitor for Virtual Machines". 2009 First International Workshop on Education Technology and Computer Science

[4] Junghwan Rhee ,Andrzej Kochut, Kirk Beaty "DeskBench: Flexible Virtual Desktop Benchmarking Toolkit". IBM T.J. Watson Research Center

[5] Goran Martinović, Josip Balen, Snježana Rimac-Drlje "Impact of the Host Operating Systems on Virtual Machine Performance". MIPRO 2010, May 24-28, 2010

[6] Kejiang Ye, Jianhua Che, Xiaohong Jiang, Jianhai Chen, Xing Li "A Performance Benchmarking Framework for Virtualization Environments". College of Computer Science, Zhejiang University

[7] Fabr´ıcio Benevenuto1, C´esar Fernandes1, Matheus Santos1, Virg´ılio Almeida1,Jussara Almeida1, G.(John) Janakiraman2, Jos´e Renato Santos2 "Performance Models for Virtualized Applications". HP Labs.

[8] S. Sudevalayam, P. Kulkarni "Affinity-aware Modeling of CPU Usage for Provisioning Virtualized Applications". February 11, 2011.

[9] Giacomo Mc. Evoy1, Patricia A.P. Costa1, Eduardo L.M. Garcia1, Bruno Schulze1 "Parallel Numerical Simulation Optimization in an Heterogeneous Environment with Virtual Machines".

[10] Paul Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauery, I. Pratt, A. Warfield." Xen and the Art of Virtualization "University of Cambridge Computer Laboratory, SOSP, 2003

[11] M. Yehuda, J. D. Mason "The Xen Hypervisor and its IO Subsystem". IBM Haifa Research Lab, IBM Linux

[12] Ulrich Drepper "Virtual Memory". October 9, 2007.

Author

Praveen G
M.Tech ,Computer science
VIT University, India
Praveen1600@gmail.com