Performance Evaluation of NAS Parallel Benchmarks in Hypervisor using Design of Experiment

Bhukya Devi Prasad* and A L Reeta Sony

1Council of Scientific & Industrial Research, New Delhi, India
2National Law University, New Delhi, India

Received 07 January 2013; revised 26 October 2013; accepted 26 November 2013

Server Consolidation is desired in order to make efficient use of available servers. This is achieved by using the concept of hypervisor. A large number of hypervisors are available in the market. Users have to choose a hypervisor depending on the type of application. Due to the large difference in the architecture of various hypervisors, it is very difficult to find which hypervisor will be best suitable for the application. The aim of this paper is to benchmark the hypervisors and examine the performance of NAS Parallel Benchmarks. Two hypervisors i.e. Xen and VMware were identified and their performance was measured against various factors.

Keywords: Xen, VMware ESX Server, NAS Parallel Benchmarks, DOE.

Introduction

Virtualization a single host machine for a number of operating systems and providing each operating system with the resources such that the functioning of one operating system does not affect the other. Here virtualization software, called as hypervisor is used and thus virtualization is implemented using software techniques with hardware assisted virtualization i.e. Intel-VT or AMD-V technology. This software provides the virtual machines with the physical resources. It also manages the scheduling and allocation of memory of physical resources among various virtual machines. These hypervisor provides each guest operating system with Virtual Machine Monitors (VMM) 1. Hence a VMM manages a single guest operating system whereas a hypervisor manages a set of VMMs. These VMM is like a virtual machine or virtualization layer for a guest operating system. This layer provides virtual hardware and virtual BIOS for the guest operating system. The virtual hardware includes virtual CPUs, memory, Network Interface Card (NIC), disk etc., thus each guest operating systems is provided with its own virtualized hardware. As a result, failure or errors of one guest operating system never affects the working of another operating system on the same machine.

As we are using x86 architecture in our experiment, it can be virtualized in three ways, Full virtualization using binary translation, Para-virtualization and Hardware-assisted virtualization. For our experiment we are using full virtualization using binary translation technique. As x86 has ring architecture consisting of four rings, of which Ring0 is given the most privilege. The VMM sits on Ring0, guest OS on Ring1, Drivers on Ring2 and user applications on Ring3. Hence VMM interacts directly with hardware and acts as the most privileged layer. Here operating system instructions are translated into binary and sent to the VMM. Again hypervisor interprets those converts into the binary and passes on to the hardware. For utilizing CPU cycles effectively or to reduce the idle time of a CPU, clusters and parallel programming concept has gained a lot of popularity. In case of clusters, where jobs are distributed among the cores, load-balancing can be achieved. The developers are concentrating more on writing parallel program code which can create process which runs independently. These processes are executed concurrently on a multi-core architecture. Thus, parallel programs are expected to improve the CPU utilization. In our experiment, we are using NAS Parallel Benchmarks (NPB), which is used to estimate the efficiency of high performance computers. We are going to create a virtual environment using the hypervisor software, install two guests OSs on them, create a virtual cluster of the guest OSs and run the benchmarks in this cluster and evaluate the performance.
Overview of Hypervisors

There are two types of hypervisors, Hosted\textsuperscript{9} and Bare-metal\textsuperscript{10}. This classification is done depending on their interaction with the hardware. In hosted, hypervisor is installed on the host operating system whereas in bare-metal, hypervisor interacts directly with the hardware. As in bare metal the hypervisor interacts directly with the hardware it is expected to perform better than the hosted. In our experiment, we are going to use two different bare metal hypervisors, Xen and VMware ESX Server.

Architecture of Xen

Xen being a bare metal hypervisor sits directly on the hardware\textsuperscript{11, 12, 13}. It runs as the most privileged layer and has got access to all the hardware directly. It manages CPU scheduling and memory management between various virtual machines. But the I/O programs like managing networks, external storage devices, video etc., are not handled by the hypervisor. These jobs are handled by a special guest operating system known as Domain 0. This is the first guest operating system installed on the hypervisor. This virtual machine has got special privileges over others. The other guest operating systems installed on Xen are considered as domain U. As the I/O jobs are handled by domain 0, domain U has to be dependent on domain 0 for their I/O work. The communication between these virtual machines is possible using event channels and page-sharing.

The communication between these virtual machines is handled by the hypervisor. Whenever a domain U has an I/O job to be done, it interrupts the hypervisor and in turn the hypervisor interrupts the Domain 0. This communication between the virtual machines for the I/O operations leads to extra overhead. Thus domain 0 is heavily loaded with work. The main problem arises when the communication and synchronization between the applications is more.

Architecture of VMware

VMware ESX Server is also a bare metal hypervisor thus is runs as the lowest and the most privileged layer\textsuperscript{14} and it has got access to all the hardware directly. Here the hypervisor manages all the jobs like CPU scheduling, memory management as well as handling the I/O request of the various virtual machines. Thus, there is no need of communication between the virtual machines. The hypervisor basically consists of two main components, Console Operating System (COS) and VM Kernel\textsuperscript{15}. The COS manages the physical world whereas the VM Kernel manages the virtual world. The COS allows user interaction with the server and it allows us to change the environment. The COS allows us to copy host files using ftp, provides us web based management console, secure shell access etc., it is also used to manage proc file system authentication and a no of applications. COS acts as bootstrap for the VM Kernel. Once VM Kernel is loaded it takes control of all the hardware, even the COS is reloaded and it runs as one of the virtual machine on VM Kernel. It provides the virtual machine environment needed for running guest operating system.

It is VM Kernel that manages the guest operating systems. It handles the scheduling of resources among the guest operating systems. It manages interaction between the virtual machine hardware and physical hardware. These hypervisors are selected as their kernel configurations are same. VMware ESX 4.0 and Citrix Xen 5.5 are used to configure the virtual machine (VM).

Virtual Cluster

Virtual Cluster\textsuperscript{16} is a computing cluster of two or more virtual machines. The virtual machines can either be on the same host machine or different host machine. In our experiment, we are installing two virtual machines on the same host machine. Here each virtual machine is provided with its own unique IP address by the hypervisor. As both the virtual machines are on the same host machine where high availability cannot be implemented. But due to the multi-core architecture load balancing is achieved.

Virtual CPU scheduling in case of Virtual Cluster

In a virtual machine, the processes are scheduled across virtual CPU’s and the virtual CPU’s are mapped to their respective physical CPU. Both the hypervisors Xen and VMware ESX Server uses the partition queue model\textsuperscript{17} for scheduling, where each virtual CPU maintains its own process queue. The jobs scheduled by that virtual CPU run only on the mapped physical CPU. But in case of virtual cluster, the jobs of a virtual CPU are distributed among all the available cores. This improves the performance and utilization of the physical core. Without a cluster the processes of a particular virtual CPU will run only in the mapped physical CPU. But in a virtual cluster the processes are distributed across the physical cores. This leads to load balancing where a particular CPU is not heavily loaded.
Testing Environment
In our experiment, we have Intel Core 2 Duo at 2.93GHz clock frequency. It has got Intel VT technology enabled in it. But it doesn’t include support for Extended Page Table (EPT). It has got a L1 Data Cache of 32KB for private data of each core along with a L1 Instruction Cache of 32KB for instructions. It is provided with a L2 cache of 3MB that is also used for fast data access. We use a SCSI disk of size 300GB with a RAM of size 4GB and each virtual machine is allocated a 2GB of RAM and 30GB of hard disk space. The evaluation of NPB is done by job completion time for different inputs for different programs on different platforms.

Host OS and Guest OS
In our experiment, we are comparing Xen5.5 and VMware ESX Server 4.0 because both of them have got the same Linux kernel i.e. 2.6. In order to form a virtual cluster we installed two guest operating systems (OS). Both of them have Fedora 8 as the operating system, which has got Linux 2.6.23.1-42.fc8 as the kernel. Here, we used the default scheduler of Xen and VMware ESX which schedules the virtual machines.

Benchmark
We use benchmark programs in order to assess the performance of a new hardware or software component. It depicts how a particular application performs in the hardware. In our experiment we used NAS Parallel Benchmarks (NPB) programs to benchmark the hypervisors.

NAS Parallel Benchmarks
The NPB are a set of complex parallel programs that have got a great deal of synchronization and communication between the threads. We used LU benchmark program, which has large number of small communications. These programs were designed to benchmark the highly parallel machines like super computers, High Performance Clusters (HPC) etc.; these benchmarks have been implemented using three programming paradigms, MPI, OpenMP and Hybrid of MPI. These benchmarks are used for CFD applications which are divided as five kernel benchmarks and three pseudo applications. The three pseudo applications include three families of programs, LU, BT and SP. These programs are written in C or FORTRAN. In our experiment we are going to run the MPI and MZ implementation of benchmarks. We are evaluating the performance of LU programs in both the implementations.

Benchmark Results & Analysis
In our experiment, we create a virtual environment using the hypervisor and analyze the performance of NPB implemented using MPI, Open MP and hybrid of MPI, the type LU on a virtual cluster. By this we compare the efficiency of these programs in Xen and VMware ESX Server.

Experimental Design and Set-up
In our experiment, we are using three factors, the type of hypervisor, benchmark and the number of virtual CPU’s as shown in table 1. Minitab offers various design options like Placket-Burman, Full-factorial and Fractional-factorial designs. The design we choose in this paper is fractional factorial. In this project total numbers of runs are 48. The experimenter will be provided with an option of selecting techniques like replication, randomization and blocking in the experiment to reduce the experimental errors. The next stage is to conduct the experiment. In this stage, the experimenter runs each test case and record the performance metrics of each test case.

The analysis of the performance metrics of each run are now noted in the Minitab tool. For analysing the data obtained during the experiment we first perform the pre-processing of the data. After applying the technique randomization high standard deviations in the output is expected, which may cause maximum variance in the data. Pre-processing helps in understanding the outliers and factors destabilizing the performance in the data. Outliers are nothing but the unusual observation in the experiment. The next step to analyse the data is residual analysis.

Residual is the difference between the actual value and fitted value. The residual analysis can be done by checking for the normality, by checking for pattern in residual vs. Fits plot, by checking histogram plot and by checking for pattern in the residuals vs. run order using four graphs. The check for the normality is identified by the normality graph by a straight line in

| Table 1—Description of factors and their levels |
|-------------|------------|------------|------------|------------|
| Factor      | Level 1    | Level 2    | Level 3    | Level 4    |
| Hypervisor  | VMware     | Xen        |            |            |
| Benchmark   | MPI- LU- A | MPI- LU- B | MZ- LU-A   | MZ –LU- B  |
| VCPU        | 2          | 4          |            |            |
the graph. The Check for pattern in residuals vs. fits plot is identified by the graph, it should be as randomly distributed cloud. This should not be in funnel shape, mirroring of residual values. The histogram plot should be bell shaped and the check for pattern in the residuals vs. run order is identified as the values should be randomly scattered. If the values follow any pattern this represents that there is a time dependency in the experimental runs. The normally distributed data is generally represented in the graph as a straight line. We usually check for normality to confirm that our sample data is normally distributed. If the graphs are not in the required pattern then experiment should be repeated till the all the four graphs satisfy the condition. We evaluated the performance of MPI and hybrid of MPI benchmarks with LU program in Xen and VMware ESX Server. We explore the change in performance due to the change in number of virtual CPU’s. We install the hypervisor on the host machine and then we install the virtual machines on them. The host file configuration of both the virtual machines is changed in order to form a a virtual cluster. Now in order to create an environment for running MPI implementation of NPB, we installed OpenMPI first. This OpenMPI is used to provide the run time environment for the MPI programs. MPI executes a single process at a time even in multi-core environment in order to use CPU to the maximum. After that, we run second benchmark i.e. NPB-MZ and it is known as multizone and also combination of MPI and OpenMPI.

**Main Effect Graph**

ANOVA stands for Analysis of Variance. It is a statistical methodology\(^{27}\) that is used for investigating data by analysing the means of subsets of data. It is used for generating the main effects graphs as shown in fig 1. The main effect graph is used to analyse the affect of an independent factor which is dependent on other independent factors and by taking the average of those independent factors.

**Benchmarking the Hypervisor by the DOE Interaction Effect**

The Fig.2 shows the running of the benchmarks on Xen and VMware ESX Server. It can be observed that the efficiency of class A benchmarks is better in all the cases. This is because the complexity of class B is more\(^{29}\). We observe that MZ-LU-A is best in both the hypervisors. This is because MZ was designed basically to run on clusters. Hence the efficiency of MZ is better than that of MPI in this environment. Also, the performance of MZ is better in case of VMware ESX Server whereas MPI performs better in case of Xen as compared to that of VMware ESX Server. Because MPI applications are sensitive to network latency and bandwidth\(^{29}\), in case of Xen architecture they perform similar as in the native environment.

The Fig. 3 shows that as the number of virtual CPU’s increases, the performance in both the

![Fig. 1—The main-effect graph for the factors Hypervisor, Benchmark and number of virtual CPU’s](image)

![Fig. 2—Running MPI-LU and MZ-LU in class A and B in Xen and VMware](image)
hypervisor decreases. This is because as the number of virtual CPUs increases, the overhead incurred in managing and allocating the virtual CPUs also increases. It gets more difficult to map these virtual CPUs to the physical CPUs. At some point it occurs that on increasing the number of virtual CPUs, the performance starts degrading drastically. In the experiment, we see that as the number of virtual CPUs allocated to a virtual machine is doubled, the performance degradation of VMware ESX Server is more than that of Xen.

The Fig. 4 depicts how the number of virtual CPUs affects the running of various benchmarks. We see that when number of virtual CPUs is less, the performance of all the benchmarks is better. For this, the reason is the increase in overhead of maintaining the virtual CPUs and allocating them to the virtual machines. Again in this case like Fig. 1 the performance of MZ-LU-A is best. In case of MPI-LU here, we see that initially the performance of MPI-LU-A is better than that of MZ-LU-B. This is due to the high complexity of class B as compared to that of class A.

**Benchmarking Results**

In Fig.1 the performance of MPI-LU-A is 34% better in Xen as compared to that of VMware ESX Server whereas in MPI-LU-B this difference reduces to 6%. In case of MZ-LU-A VMware ESX Server performs 40% better than Xen and in MZ-LU-B it is 15% better than Xen. In Fig.2 when the number of virtual CPU’s is two in number the performance of VMware ESX Server is better than Xen by 7%. But as the numbers of virtual CPU’s are increased to four in number, we see that Xen outperforms VMware ESX Server by 11%. In Fig. 4, in all the cases performance is decreasing with increase in the number of virtual CPU’s. We observe that the performance of MZ-LU-A is degrades by 58% as the number of virtual CPU’s are increased from two to four whereas in case of MZ-LU-B the performance degrades by 11%. In MPI-LU-A, the performance drops by a huge margin of 87% and in MPI-LU-B the drop is 72%.

**Conclusion**

In our experiment, we run various implementations of NPB on the two hypervisors and Xen hypervisor performs best in all the aspects. When we use the number of virtual CPUs as one of the factors we find that the increase in the number of virtual CPUs affects the performance adversely. And when we consider class of the benchmark as the factor we find that in all cases class A benchmarks performs better than that of class B. Among all the benchmarks, we find that MZ performs the best.

**Disclaimer**

The results described in this paper are derived from a particular benchmark configuration, created specifically for this study; any extrapolation to other product comparison may not produce the same main/interaction graphs.

**Acknowledgement**

The authors convey immense reverence and thankfulness to Prof. Ramachandram S, College of Engineering, Computer Science Department, Osmania
University, Hyderabad, India for providing test-bed, suggestions, guidance, encouragement and kind support for this paper.

References