UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science

Allan Trukits

The Cost of Virtualization for Scientific Computing

Bachelor's Thesis (6 ECTS)

Supervisor: Pelle Jakovits

Author: ..... "....." May 2013

Supervisor: ..... "....." May 2013

Allowed to defence:

Professor: ..... "....." May 2013

Tartu 2013

Table of Contents

Introduction

Virtualization is a term that refers to various techniques, methods or approaches of creating a virtual, rather than actual, version of something [19]. In general, virtualization provides means to deploy multiple operating systems on one physical computer. These multiple instances act as separate computing units and are seen as physical machines to other computers in the network. This way, virtualization provides more security and configurability. Administrators do not have to give access to two persons on the same computer, instead they create two virtual machines which are totally separate and these two persons might not even be aware of one another when they connect to these virtual machines over the internet. This way, another instance of the virtual machine is safer from the other's doings although they share the same physical resources.

In this paper's context virtualization means that the operating system and the hardware platform including the CPU, the storage device and network resources are made virtual using different types of hypervisor. These types are full virtualization, partial virtualization and paravirtualization [6].

Virtualization technology is widely used in cloud computing which offers virtually infinite resources and because of that is suitable for solving resource hungry scientific computing problems [18]. The cost of virtualization in the heading refers to performance difference rather than monetary value. Undisputedly virtualization of hardware saves money and this is not the question today. The question is how much CPU, network and disk input-output performance loss is to be expected through virtualization. This question is answered by benchmarking with the NAS Parallel Benchmarks (NPB) [7] which was developed by the NASA and is specifically designed to test parallel computing performance. This paper focuses on tests implemented in MPI [8] programming model and written in Fortran 77 and C programming language. Tests are described more thoroughly later on. As this paper focuses on the cost of virtualization for scientific computing then these tests are most suitable because they represent a large problem which is split into large number of jobs which are then run on all computers in the cluster. Because of that it is important to know the impact on the network performance because in the case of scientific computing it is often necessary for jobs to communicate with one another. This form of computing is called parallel computing. It is also necessary to measure disk input-output performance change because it is sometimes needed to parse large sets of data files. Last but not the least, we are interested in general impact on the computing power thus CPU and memory tests are due. Another benchmarking tool used in

this thesis is Phoronix Test Suite (PTS) [11] which is designed to test the performance of a single computing unit. This is all covered in more detail in the next chapter.

In the case of running multiple number of virtual machines on one host, results should not come as a surprise - more virtual machines on one host concurrently trying to solve some puzzle will take more time and vice versa. This paper also tests scalability of these systems because sometimes it is necessary to take x number of computers any set up y number of virtual machines on them which are then used for different jobs. It makes sense for ease of use and security purposes to set up many virtual machines but these different jobs might not be run concurrently. Thus this paper tests the performance of four virtual machines on one host, four virtual machines on two hosts, four virtual machines on four hosts and some other combinations.

In addition this thesis finds a comparable Amazon EC2 [20] instance to two servers in university's possession. This should provide interesting insight whether it is useful for the University of Tartu to use Amazon services or set up its own cloud. As Amazon does not fully reveal their hardware specifications and they also deploy multiple instances on one machine, it is very hard to find comparable hardware but using the same testing methods as for comparing Xen to KVM, it is possible to find close enough instance that matches by the processing power. This info is particularly interesting to Mobile & Cloud Computing Laboratory [16] of the University of Tartu.

The first section of this thesis brings out previously done related works and points out what they discovered and what their results were. Also the state of the art is described in that chapter which covers the benchmark and virtualization technologies used in this paper.

The second chapter of this thesis describes how to set up the same environment to be able to replicate the results delivered in this thesis. That chapter also exposes which configuration challenges and problems are to be expected.

In the third section the analysis of the benchmarks' results is brought out and conclusions are drawn from the analysis.

In the last section of this paper we are matching an Amazon EC2 instance to the university's server and analyzing benchmark data to see if it is more profitable to run scientific calculation on Amazon instance or on the university's server.

We are expecting Xen and KVM to be more equal in performance than in the past but the test results will show whether expectations will meet the reality.

1. State of the Art

1.1. Xen and KVM

Xen [4] is freeware hypervisor software which enables us to virtualize physical resources. In this paper's tests Xen is used in paravirtualization mode which means that the hardware environment is not simulated but guests are run in a modified operating system.

Another type of virtualization used in this paper's benchmarks is full virtualization for Kernel-based Virtual Machine (KVM) [5], also a freeware hypervisor, which means that almost all hardware is simulated to allow software to run unmodified. In this case KVM also uses hardware-assisted virtualization with full virtualization to enable more efficient use of processor power.

These should be the best performing options for Xen and KVM to get the best out of the two most popular freeware virtualization software.

1.2. Related Works

There are many previous studies related to just testing Xen or KVM separately or just analyzing virtualization more generally but there are only a few published works which have benchmarked both Xen and KVM and provided a clear comparison between the two hypervisors for the scientific computing tasks.

Todd Deshane et al. [1] also used The Phoronix Test Suite to measure performance levels on a single computing unit i.e. the host, the Xen guest or the KVM guest. They also performed scalability tests but they did it quite differently than we. They deployed n numbers of guests on one host and run the same tests concurrently whereas we are running specialized parallel computing tests which are written specifically to test parallel computing performance by splitting one job between all of the guests, not running the same job concurrently. Nonetheless, this approach showed that KVM did not perform very well when too many guests (with four guests, one crashed; with 16 guests, seven crashed) were running on one host. A significant percent of guests just crashed while Xen was able to handle multiple guests very well (with 30 guests, none crashed), only showing increase in testing time which is expected. Overall this article concluded that Xen performed very closely to the host and that KVM had a little more degradation on almost all of the tests performed except in the case of read and write to disk test which may have been due to the disk caching capabilities. They used Xen 3.2.1 and KVM 62 and this article was published in the June of 2008.

Andrea Chierici et al. [2] very briefly and understandably explain what Xen and KVM are about and what types of virtualization they use. They make a good case in qualitative test for both KVM and Xen as they switched from Xen to KVM on their own systems which were used for real applications for many users and did not see any noticeable performance changes. But when they carried out quantitative tests it became clear that KVM did not quite perform as well regarding network and disk input and output performance, although CPU performance showed similar results. For scalability they used the same approach as in the first article. It seems that they did not observe the problem of KVM to run a multiple number of guests which is probably due to the fact that KVM had been developed further as they used a later version of it. They used Xen 3.2.1 and KVM 83 and this article was published in 2010. So in two years some progress was made in the stability of KVM but some drop in disk I/O performance was noted. As this was more than three years ago perhaps the performance of KVM has improved by now.

Lucas Nussbaum et al. [3] start with providing an interesting point - that many processing jobs do not fully take advantage of the multicore architecture. Nowadays I cannot imagine processing units with just one core, even our smartphones have two and some have even four cores. Thus deploying virtual machines per core would provide an easy way to share physical resources among several jobs. This article also compares different types of virtualization i.e. paravirtualization vs. full virtualization. Paravirtualization works better on all test cases and this is the technique that this paper is covering. Similarly to the second article they show that CPU performance is very even and that KVM did not perform as well as XEN regarding disk I/O and networking tests. For scalability test they used the same approach as the other two articles but also did something similar to what we are going to do. They used HPC Challenge benchmarks which are developed to measure the performance of parallel computers. Again Xen performed better in most cases except in the network throughput. They used Xen 3.3.1 and KVM 84 and this article was published in the October of 2009.

These articles and this thesis are very similar in nature but our approach is different as we are testing the scalability of virtual machines as well as physical machines and we are using different benchmarking tools and as some time has passed, all of the results above might not be relevant anymore, so this work is necessary to stay informed of the differences and similarities between these two freeware virtualization platforms.

1.3. NAS Parallel Benchmarks

NPB tests are specifically designed to test the performance of distributed computers which is what scientific computing on many occasions rely on because of vast and cumbersome algorithms that need a lot of computing power which regular computers cannot output. Problem sizes in the NPB are predefined and indicated as different classes. Two of these classes are used in this work. Two tests are of B class and one is of C class. At first we thought all of the tests to be of class C, but class C proved to be too expensive in memory and hard drive usage for two tests so the class was lowered for those two benchmarks.

Three tests were chosen, each representing a different problem and thus measures different aspects of performance cost. One of them is Integer Sort (IS) Benchmark which tests a sorting operation which is used to reassign particles to the appropriate cells. This is used in particle-in-cell applications of physics. The implementation is based on a bucket sort [14]. The benchmark tests integer computation speed while floating point arithmetic is not involved thus this benchmark relies on CPU performance, but also a significant amount of data is transferred and thus it also tests network capabilities [12]. Class B was used for this test which means that the number of keys to generate is 2^{25} while the maximum value of the key is 2^{21} [13]. Keys are the elements that are going to be sorted.

Another test chosen was the Embarrassingly Parallel (EP) Benchmark which solves a problem typical to many Monte Carlo applications that is accumulating two-dimensional statistics from a large number of Gaussian pseudorandom numbers. This benchmark requires almost no data transfer, except in the beginning and in the end of the test, and in some sense it provides an estimate of the upper achievable limits for the floating-point performance [12]. The calculation also contains a significant number of logarithm and square root operations [14]. Class C was used for this test which means that the amount of random number pairs generated was 2^{32} [13].

The third test chosen was the Block-Tri-diagonal Input Output (BT-IO) Benchmark which by the name of it tests I/O performance. Class B was used because class C test requires writing 6.8 GB of data while the size of the virtual HDD was only 5 GB. Class B benchmark writes 1.7 GB of data onto the disk. [13, 15]

1.4. Phoronix Test Suite

The Phoronix Test Suite (PTS) is a freeware benchmarking package developed to test a single hardware unit. It cannot be run like NPB to split the same job between many machines but as NPB is designed to do just that then NPB might not show meaningful results when running on one virtual or actual computing unit. For that reason I have chosen the PTS - to compare one actual host machine against one virtual machine. Again three benchmarks were chosen from the Phoronix Test Suite to test different aspects of performance.

Firstly, a test named Primesieve of version 1.1.1 was chosen to test CPU's L1 cache performance. Phoronix does not get into more detail about their tests but just as in the case of NPB, the whole suite is open-source so anyone could read the source code to see what exactly is done during testing.

Secondly, a test named Stream was chosen which was of version 1.1.0 and it is designed to test system memory performance. This test consists of four parts: copy, scale, add and triad. Each of them perform different tasks. Copy test just takes something that is in some memory address and writes it into another memory address (a(i) = b(i)). Scale test takes something that is in some memory address, multiplies it to something and then writes it into another memory address (a(i) = q*b(i)). Test named add takes two values from memory, adds them together and writes them to another memory address (a(i) = b(i) + c(i)). Triad test takes all three previous tests and puts them together, it takes some value from one memory address, multiplies it to something, then takes another values from memory and adds it to the product and then writes the results into another memory address (a(i) = b(i) + q*c(i)). [24]

Last but not the least a test named Unpacking the Linux Kernel of version 1.0.0 was picked to test disk input-output performance by measuring how long it takes to extract the .tar.bz2 of the Linux kernel package.

1.5. Amazon Elastic Compute Cloud 2

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers. [20]

Amazon is a very popular and useful service to use for scientific calculations as one could just order exact amount of resources on-demand. For that reason this paper is trying to see if and in which situations it would be beneficial for the Mobile & Cloud Computing Laboratory [16] of the University of Tartu to use Amazon's resources instead of the already existing servers in

the university's possession because ordering these resources costs money and if there would not be any substantial benefit over university's servers then there would not be any reason to use them.

Amazon uses Xen's paravirtualization technology for its instances.

2. Environment Configuration

In this section a lot of code segments are brought out due to the lack of a single step by step tutorial on the internet that one could follow to replicate the results of this thesis. So in a way this thesis provides a new tutorial to set up a cluster of virtual machines for parallel computing but unfortunately as all of the other tutorials in the internet will be outdated soon because of the level of detail that is put into versions of different components.

2.1. Hardware

Four identical computers were used. All equipped with Intel Core 2 Quad Q9500 processors running at 2.83 GHz. As the name would suggest these are four core processors. So in total we had 16 cores to use. Hardware virtualization support was turned on from the BIOS.

All machines had 6.9 GiB of RAM to use but the amount of memory does not matter in these tests as much as the speed of the memory because these tests do not consume much memory, they only test the speed of it so 1 GiB of RAM would be sufficient enough. These were DDR2 memories with 400 MHz clock speed and 800 MT/s transfer rate.

500 GB Western Digital WD5000AAKS-0 hard drive was used for these tests. This is a 7200 RPM 16 MB cache hard drive with the 3.0 Gb/s SATA 2 interface. Operating system was installed on the ext4 file system.

Network was built on 100 Mbit speed connections through a switch.

2.2. Software

Ubuntu 12.04.2 LTS (precise) 64 bit version with the 3.2.0-38-generic Linux kernel. Mpich2 [9], gfortran [10] and ‘make’ are required to compile and run NPB benchmarks. PHP 5.0 is required to run the Phoronix Test Suite [11].

2.2.1. Networking

Network configurations should be done prior to KVM and Xen guests installations otherwise there will be bridging difficulties and guests will not receive connections correctly. Also when bridging is done forehand then creating guests is pretty much enter-one-command operation. To create a bridge one must first configure primary Ethernet adapter to receive its IP address manually and then create a bridge that will take an IP address from a DHCP server. The following configuration works:

```
sudo apt-get install bridge-utils
sudo nano /etc/network/interfaces
        auto eth0
        iface eth0 inet manual

        auto br0
        iface br0 inet dhcp
                bridge_ports eth0
                bridge_stp off
                bridge_maxwait 0
                bridge_fd 0
sudo /etc/init.d/networking restart
```

2.2.2. KVM

The version of KVM is determined by Linux kernel which was 3.2.0-38. To get the best results out of KVM, hardware virtualization must be enabled from the BIOS if the processor supports it. In this case it does. To check whether the system is currently using hardware virtualization or not, one should run the following command:

```
kvm-ok
```

The following command installs KVM:

```
sudo apt-get install qemu-kvm libvirt-bin ubuntu-vm-builder virt-manager
```

To install KVM guest run:

```
sudo ubuntu-vm-builder kvm precise --domain Ubuntu4 -d Ubuntu4 -a amd64 --hostname Ubuntu4 --mem 1024 --user scicloud --pass scicloud --cpus 4  --components main,universe --addpkg openssh-server --addpkg nano --addpkg make --addpkg wget --addpkg unzip --addpkg mpich2 --addpkg php5 --addpkg php5-gd --addpkg gfortran --bridge br0 --libvirt qemu:///system ;.
```

This results in KVM virtual machine or guest that is getting access to all four cores of the host machine, 1024 MB of memory and a 5 GB virtual Qemu hard disk drive with the ext4 file system. It runs the same Ubuntu 12.04 operation system with the 3.2.0-38 64 bit kernel as the host machine.

2.2.3. Xen

The version of Xen used was 4.1.2. Installing Xen is more complicated and time consuming than installing of KVM. Firstly, one should install Xen hypervisor and required components using the following command:

```
sudo apt-get install xen-hypervisor-4.1-amd64 xen-tools
```

Secondly, after restarting the computer and running the Ubuntu with the Xen modified version of the kernel one should add a symbolic link for xen-tools because the latest Ubuntu

release configuration file is missing and without it guests will not be built. To add a symbolic link run:

```
cd /usr/lib/xen-tools
sudo ln -s karmic.d precise.d
```

Then it is necessary to replace all text from `/etc/xen-tools/xen-tools.conf` file with the configuration in the appendix 1. Xen guest builder can also be constructed similarly to the KVM guest builder but it is not possible to configure the number of CPUs given to the guest with that builder so one must use this configuration file instead and when that is configured then run:

```
xen-create-image -hostname=Ubuntu4
```

When creating a Xen guest, a password cannot be given in advance, on the contrary to the KVM guest creation; instead it is asked during the installation interactively which is somewhat inconvenient. But on the other hand Xen provides an option to boot guests automatically after host system has been started, this is a very useful property to have when administering virtual machines. Another counterclaim might be that with this installation method it is not comfortably possible to make Xen guest use ext4 file system. Some sources indicate that it is somehow possible to make Xen guest use ext4 file system but this option was not further investigated in the scope of this paper.

Unfortunately Xen images come as blank as they can, meaning no SSH (on some occasions SSH was present, but not always), no text editors that could be used through SSH connection and on multiple occasions apt-get configuration files were messed up and even if SSH configuration was available then it was not possible to install anything. As fixing the broken configuration files for apt-get included using a text editor and the only one available by default is ‘vi’ then it was not possible to fix through SSH connection since ‘vi’ commands cannot be sent through one. So it is best to fully configure Xen guests while still using a terminal connected directly to it. Most importantly one should use the following command to install necessary packages to use the benchmarking software:

```
apt-get install mpich2 php5 php5-gd make gfortran
```

This build method results in a Xen guest with access to 4 CPU cores and 1024 MB of memory and 5GB of disk space on ext3 file system where an Ubuntu 12.04 operation system is ran.

2.2.4. General Configuration

To be able to use NPB and to comfortably navigate between guests and hosts it is necessary to generate a RSA key for the SSH and to distribute it to all hosts and guests. This can be done by running the following commands analogously:

```
ssh-keygen -t rsa
cat /$HOME/.ssh/id_rsa.pub >> /$HOME/.ssh/authorized_keys
scp -r /$HOME/.ssh/ username@172.17.x.y:
```

Also it is required to make every guest's host file point to the machine where the benchmark will be initiated. This pointer must be by the machine's hostname. So even if all guests share the same hostname, this shared hostname must point to the 'first' computer. This is due to the NPB sending results back to the main thread by the hostname and not by the IP address.

2.2.5. NAS Parallel Benchmarks

The version of the NPB used was 3.3.1 and as stated earlier it was a MPI package of the NPB. Note that while NPB is a freeware, it is necessary to register yourself in NASA and state your business to get the right to download this software.

Configuring the NPB and getting it to run is quite tricky at first because there is little to no information on the internet about it and the provided manuals assume you are an expert. Firstly, one should give the NPB folder full access rights as otherwise it will not build. To do this use the following command in the NPB folder:

```
chmod 755 .
```

Then it is necessary to configure NPB to start using correct Fortran and C compilers. So one should navigate to the 'config' folder and create a 'make.def' file out of the provided template and change according rows into the following:

```
MPIF77 = mpif77
MPICC = mpicc
CONVERTFLAG = -DFORTRAN_REC_SIZE=4
```

Then tests used in this paper can be compiled using the following commands:

```
make is NPROCS=16 CLASS=B
make ep NPROCS=16 CLASS=C
make bt NPROCS=16 CLASS=B SUBTYPE=full
```

If successful then one should create a text file under the `bin` folder (NPB subfolder), where all the compiled benchmarks are placed, named 'machines' for example and into that it is necessary to write IP addresses of the computers used for the current benchmark. This file will be in the form of IP addresses line by line followed by a colon after which one can say how many processes should be run on that machine otherwise processes will be deployed in a 'round robin' fashion [17]. The format of the machine file:

```
172.17.x.y:4
172.17.x.z:4
```

Finally all these compiled tests must be distributed to all of the guest and hosts that are participating in the current launch of the benchmark. And all of these files must be under the same absolute path thus the username for all of the guests and hosts must be the same.

Only now is one ready to launch the NAS Parallel Benchmarks. And to do that the following command can be used analogously:

```
mpirun -machinefile machines -np 16 ./is.B.16
```

Note that the dot and the slash before the file name are mandatory otherwise one would get error messages not even remotely describing the problem.

2.2.6. Phoronix Test Suite

The version of the Phoronix Test Suite used was 4.4.0. Installation of this software is fairly easy but it must be stated that PHP 5 must be installed beforehand otherwise Phoronix starts generating errors that the internet has never heard of and it is not anymore possible to install PHP and continue. To be clear, one should use the following commands before continuing:

```
sudo apt-get install php5
sudo apt-get install php5-gd
```

After that it is only a trouble of installing the benchmark suite using the following commands:

```
sudo dpkg -i phoronix-test-suite_4.4.0_all.deb
phoronix-test-suite
phoronix-test-suite install pts/stream
phoronix-test-suite install pts/primesieve
phoronix-test-suite install pts/unpack-linux
```

And to run the benchmarks use:

```
phoronix-test-suite benchmark pts/stream
phoronix-test-suite benchmark pts/primesieve
phoronix-test-suite benchmark pts/unpack-linux
```

3. Benchmarking

3.1. Single Hardware Object

Here we are comparing a single host machine instance to the KVM and Xen virtual machine instances. These tests show if just by adding a virtual layer between the program and the hardware is going to affect the performance in any way. All tests were performed three times and an arithmetic mean was taken to eliminate possible outliers. All instances in the following tests have access to four CPU cores.

Firstly, we are comparing CPU performance results with the Phoronix Test Suite's Primesieve benchmark. (Fig. 1) The results favor KVM, even to the host by showing 5.4% less time consumption which is surprising considering that KVM and the host use the same unmodified kernel. As KVM utilizes to its advantage the use of the hardware supported virtualization by CPU it is a no surprise that KVM performs better than Xen. But since this feature is currently supported only by CPUs and not hard disk drives or memories then it is not expected for KVM to be better at those tests.

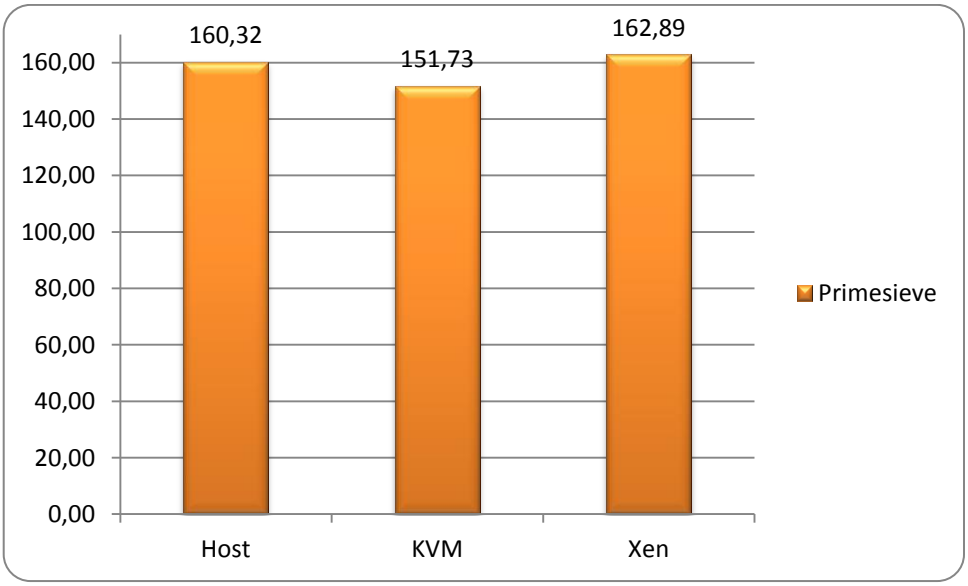

Figure 1. PTS Primesieve benchmark. Units are in seconds. Smaller value is better.

Secondly, we are comparing memory performance results with the PTS Stream test. (Fig. 2) Results are very balanced, only one to two per cent of difference which is not enough to conclude much. KVM seems to perform a little better than Xen but compared to the host one cannot really tell a difference.

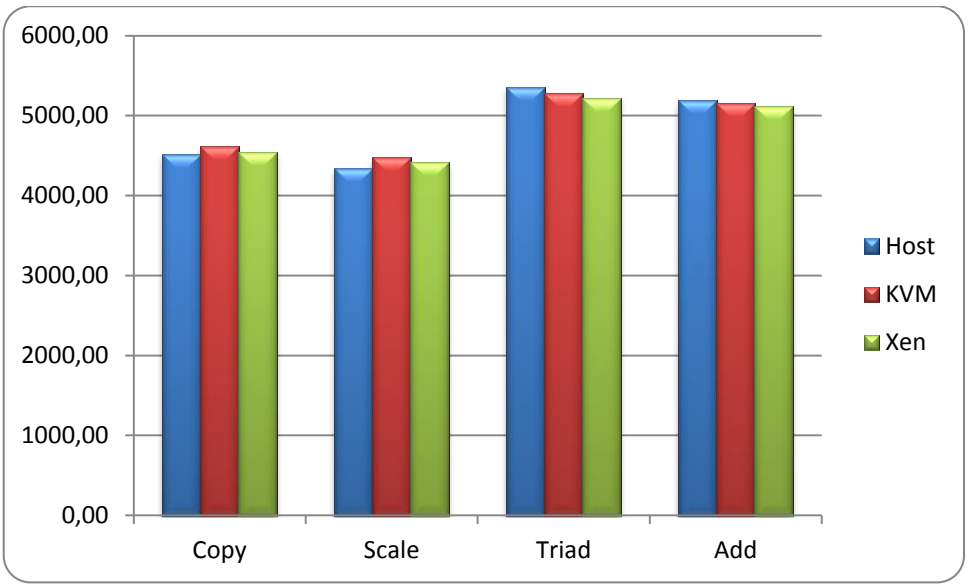

Figure 2. PTS Stream benchmark. Units are in Mb/s. Higher value is better.

Thirdly, we are looking into hard disk drive's input-output operation performance with the Unpacking the Linux Kernel benchmark. (Fig. 3) Results reveal 13-16% overhead in time for virtual machines which is a lot but also kind of expected. In comparison of KVM, Xen seems to be able to handle HDD I/O a little better.

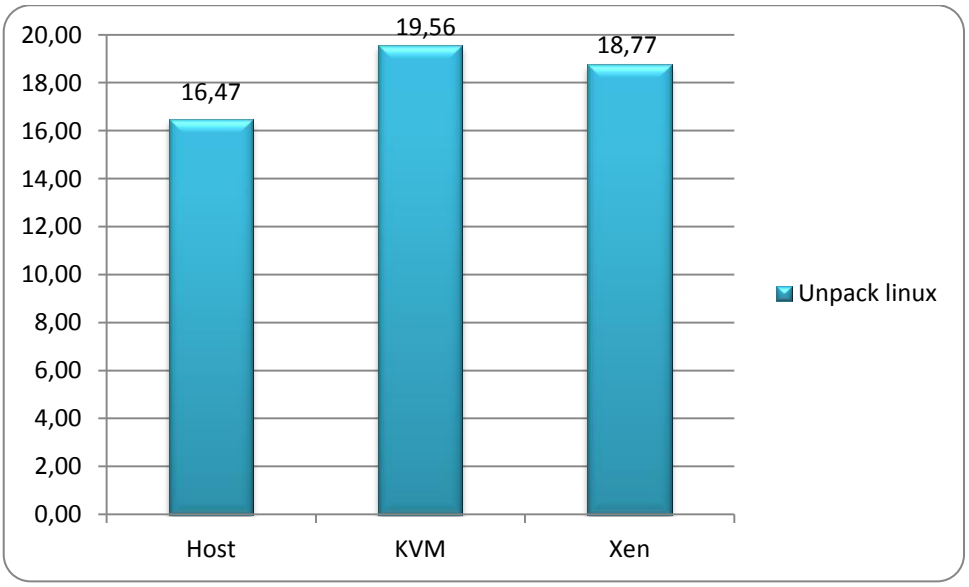

Figure 3. PTS Unpacking the Linux Kernel benchmark. Units are in seconds. Smaller value is better.

In terms of computing power, adding a virtualization layer does not seem to affect performance much but when it is necessary to write or read lots of data from disc then a significant loss in time should be taken into account.

3.1.1. Scaling the Number of Virtual Machines

Here we are comparing one, two and four instances of KVM and Xen virtual machines to one another. These tests show if it would be better to deploy more guests rather than using a single host to run an application. It is now a well-established fact that running multiple virtual machines (VM) on one host to let different people run different applications at different times, is a very effective use of resources because there are less zero-usage hours. We are trying to see if it is also reasonable to use the same approach to run the same application on all of those guests. All instances in the following tests have access to four physical CPU cores which for virtual machines means that if there are one, two or four guests running then they have four, eight or 16 virtual cores respectively. One MPI job per virtual core for virtual machines is deployed. There are always four MPI jobs deployed on hosts i.e. one job per actual core.

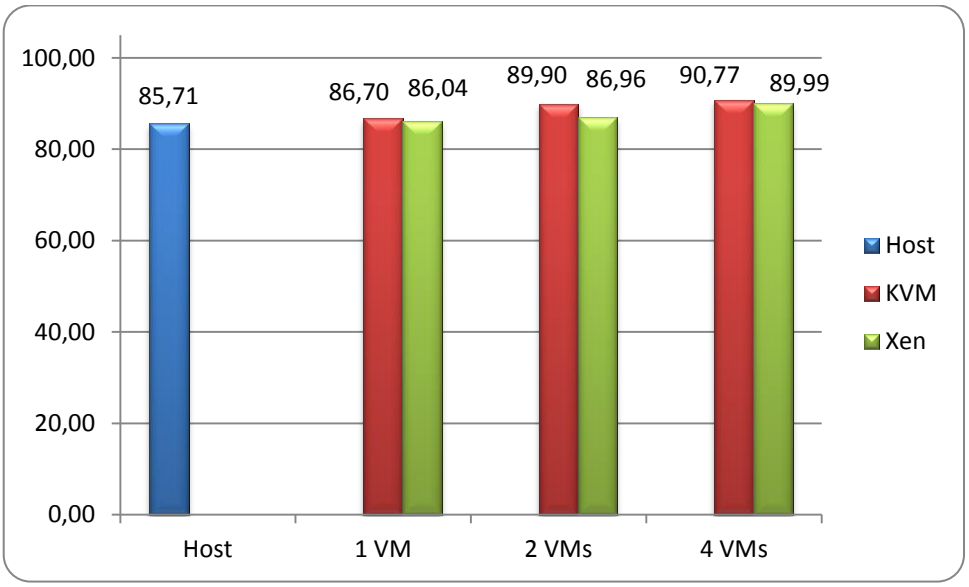

Figure 4. NPB Embarrassingly Parallel benchmark. Units are in seconds. Smaller value is better.

Firstly, results from the NPB Embarrassingly Parallel benchmark. Figure 4 indicates similar results for all of the tests. Only a 1% drop in performance when going from host to a virtual machine which supports the last chapter's conclusion that CPU performance is not affected much by virtualization. But if observed more closely then a little rise in time consumption is already notable - a 5% rise when comparing host's result to the four KVM instances' result. Heavier drop in performance should be expected when scaling the number of virtual machines any further although based on the figure 4 results it seems that KVM and Xen are behaving very similarly. But this is just only the CPU benchmark which is expected to do the best because of the hardware support.

Secondly, figure 5 represents the results from the NPB Integer Sort benchmark. This is the benchmark where networking is a large factor and it is clearly seen on the chart. Of course for

the host and a single virtual machine there is no networking involved and they are acting as test controls. Increasing the number of virtual machines is expected to increase the time consumption but already when just two virtual machine instances must communicate with each other, performance loss is staggering. It took two KVM instances 14 times longer to complete the task and for Xen it was 29 times longer. The case for four instances is even worse – 18 and 68 times longer respectively. I would like to remind that all results are arithmetic means of three separate test and all of these three tests were bearing similar results. A little rise is to be expected but this is not in a reasonable proportion. And what is more - Xen is performing extremely badly in this situation. We are concluding that 16 MPI jobs on four virtual machines trying to communicate with each other and to the main job launcher thread is too much to handle for virtual network interfaces. As we scale these tests in the next chapter, we see if the reason for it could be that virtual network interfaces emulated for virtual machines on one host cannot somehow effectively communicate with each other or does this problem also bear upon multiple hosts.

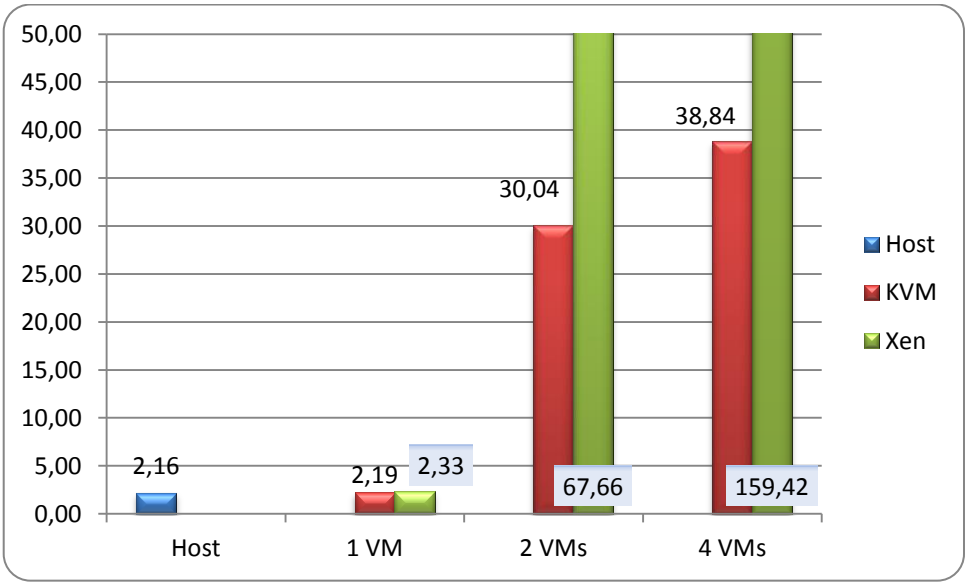

Figure 5. NPB Integer Sort benchmark. Units are in seconds. Smaller value is better.

Thirdly, we are comparing the results for the NPB BT-IO benchmark. Figure 6 is revealing that on the contrary to the Phoronix's disk benchmark's write results, Xen seems to be performing slower than KVM and something is seriously affecting the performance of the Xen VM while running tests on two and four instances. Even KVM does not seem to be doing good - taking 3.4 times more time to complete the assignment when comparing one virtual machine to two and taking 5.4 times more time when comparing to four virtual machines.

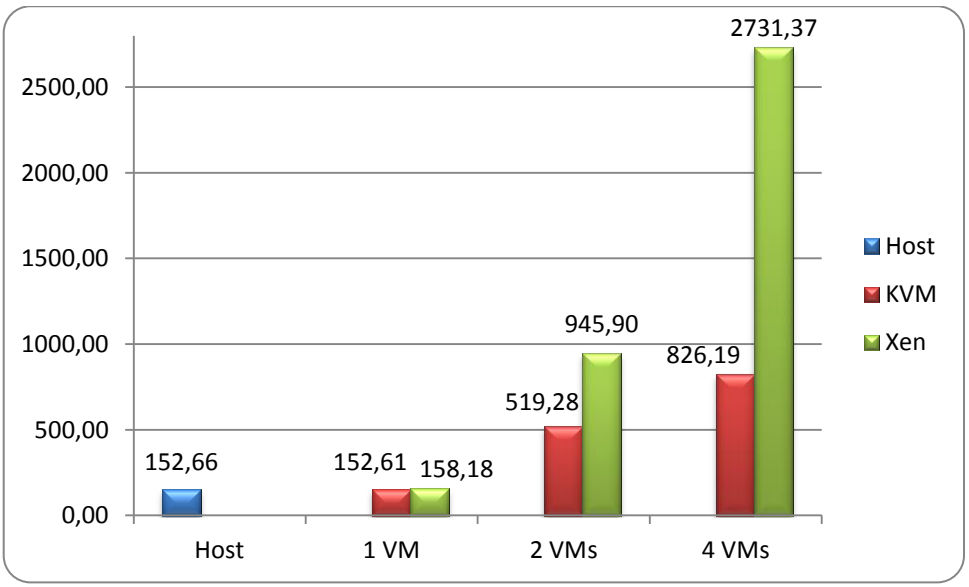

Figure 6. NPB Block Tri-diagonal Input Output benchmark. Units are in seconds. Smaller value is better.

With the previous information in mind it is safe to conclude that it is not profitable to run multiple virtual machines on one physical machine to perform concurrent calculations but when just running multiple virtual machines and letting different people use them on different times of a day, thus keeping the physical machine busy at most times, is not affecting the performance of a single virtual machine much or none at all. Xen's virtual machines' trouble to cope with multiple guests communicating with each other on one host and writing data onto the same physical hard drive concurrently is an interesting topic to look into in the future.

3.2. Multiple Hardware Objects

In this section we are comparing one host with four virtual machines to two hosts with four virtual machines to four hosts with four virtual machines with both KVM and Xen. These test show how previous results scale to a larger number of computers than just one. In previous tests there were always just four physical cores available to do the calculations but here this is first scaled to eight cores and then to 16 cores. Both KVM and Xen clusters have 16 virtual cores in all tests, only the number of underlying physical number of cores changes, ergo 16 MPI jobs are launched for all of the following benchmarks. We are also comparing all of these results to physical machine results. As it might be more performance-wise to just scale the number of host machines and not use the virtualization at all. This is of course more costly but it should give better performance results. Host tests are not run by making 16 jobs for all tests but making four, eight and 16 jobs whether one, two or four machines are used respectively.

3.2.1. Scaling the Number of Physical Machines

Firstly, the NPB EP test. Figure 7 shows how this benchmark scales very well as adding more physical machines to the cluster improves the performance in an expected way. This CPU performance test again confirms that Xen and KVM are very equal when just using computing power and only marginal communication is held between virtual machines or when no disk input is required. Looking at four machines test results where there is only a single layer of virtualization added for the virtual machines i.e. one virtual machine per physical machine; it is seen that performance times are very close, meaning that adding a virtual layer will not affect the computing performance in fact we would have expected the time for the host (23.28 sec) to be better than the time for the virtual machines (22.33 sec and 22.8 sec) but it seems to be an anomaly and should correct itself when an infinite number of tests are run.

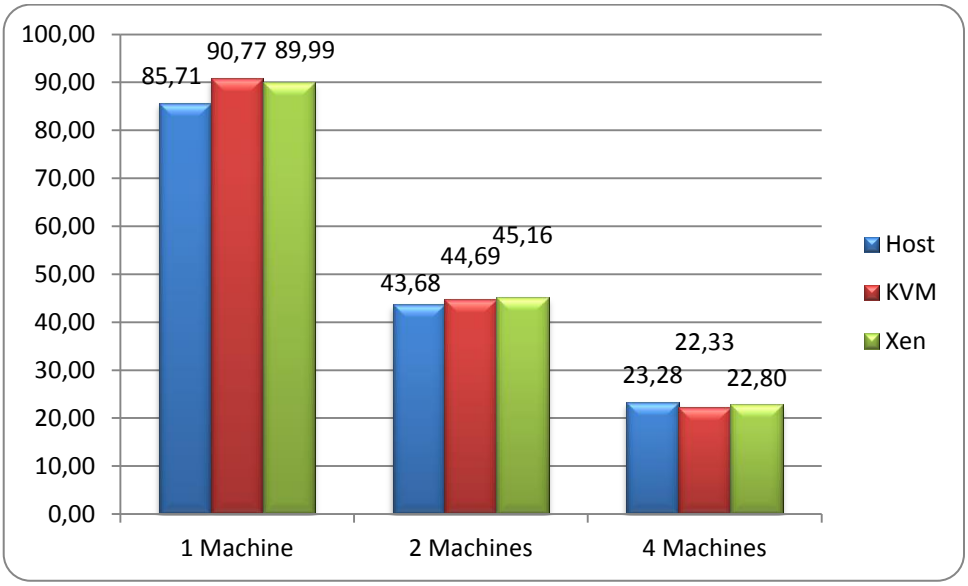

Figure 7. NPB Embarrassingly Parallel benchmark. Units are in seconds. Smaller value is better.

Secondly, we present the results from the NPB IS benchmark. Fig. 8 is depicting the results which are quite conclusive - KVM is performing much better than Xen. Comparing virtual machine results to the host result, it is apparent that networking between virtual machines is an issue for both Xen and KVM but Xen also performs 2.45 times worse than the KVM or the host on four machines where there is one virtual machine per physical machine which indicates that Xen's guests' virtual network interfaces have serious troubles communicating with each other.

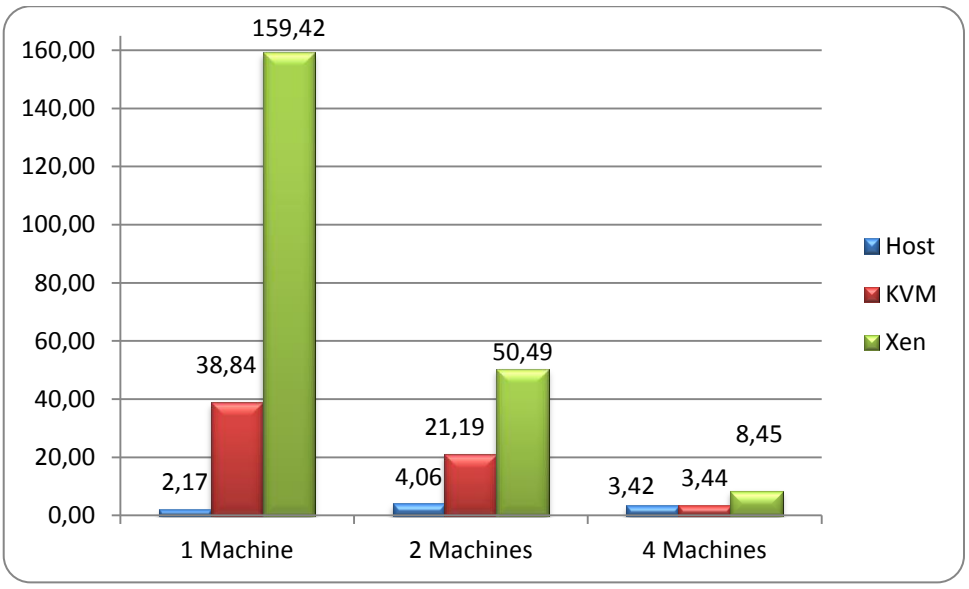

Figure 8. NPB Integer Sort benchmark. Units are in seconds. Smaller value is better.

Thirdly, we are comparing the results from the NPB BT-IO benchmark. Figure 9 is representing the collected data. It is clearly seen that when more sources are trying to write data onto the disk then the HDD cannot keep up and writing the same amount of data takes considerably longer. KVM is doing much better in this test than Xen, just like in the previous BT-IO test and while other benchmarks seem to support the idea that adding a virtualization layer will not affect performance very much then this test agrees with the Phoronix's Unpacking The Linux Kernel benchmark that an extra layer does affect disk related operations very badly. It took both KVM and Xen twice as much time to write 1.7 GB of data onto the disk when comparing the four machine results.

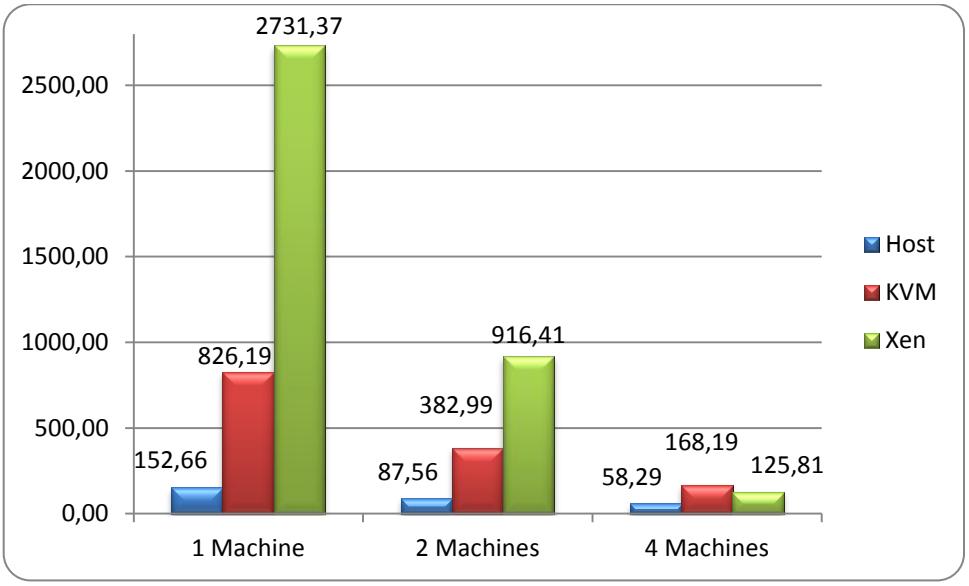

Figure 9. NPB Block Tri-diagonal Input Output benchmark. Units are in seconds. Smaller value is better.

Again it would be interesting to research more into why Xen performs so poorly in networking and hard drive I/O benchmarks. It would also be interesting to see the BT-IO

results when HDDs would be replaced with the SSDs which do not have that large of a problem writing data concurrently and are much faster than HDDs in general.

To conclude this chapter we have to point out that KVM seems more feasible option to choose when going for the virtualization since almost on all cases KVM performs a little or much better than Xen. It is also necessary to bring out that adding a virtualization layer alone does not affect the performance at all or just a little but when there are large quantities of data to write or send over the internet then one really needs to weight pros and cons.

4. Amazon EC2

In this section we are matching Amazon EC2 instances to the university's servers. We do it by running the Phoronix Test Suite's Primesieve benchmark which tests CPU performance. We believe that CPU is the property to compare by because it is nearly impossible to match all parameters – RAM, CPU, hard drive I/O and networking and CPU is the core of the computing power and it is what matters the most in performance. What is more, RAM can easily be upgraded and so can be a hard drive. After we have found comparable instances by CPU we launch the NPB Integer Sort benchmark on a single machine and in two computer cluster to test the networking loss or gain in the Amazon cloud.

It is useful to see which is the comparable instance by CPU performance as this is what scientific algorithms mostly require and after running cluster tests we see if it is better to make a cluster of university's servers or whether it is more useful to deploy algorithms on the Amazon cloud.

4.1. Specifications

The university's server has a Xeon E5606 CPU running at 2.13 GHz clock speed and it has four cores but it is running on hyper threading thus making it eight cores. The operating system is Ubuntu 12.04.1 LTS (precise) 64 bit version which is virtualized by Open Stack cloud [21] technology using KVM virtualization. The virtual machine on that machine is made to fully occupy the physical hardware meaning it has access to all eight cores.

For the first estimation we chose Amazon's 'M3 Double Extra Large Instance' (m3.2xlarge) which has a Xeon E5-2670 running at 2.6 GHz clock speed and it has eight cores but it is not using hyper threading thus the number of cores is not doubled. Since amazon does not provide this information it was gathered by running the following command in the Ubuntu 12.04.1 LTS:

```
cat /proc/cpuinfo
```

This provides accurate information about CPUs and how many we have access to. Some knowledge and expertise from Huan Liu's article [22] was used to determine the best candidates to match our server.

The 'M3 Double Extra Large Instance' has a much better CPU than our server but as this is instance does not take up the whole physical machine then there are probably some more virtual machines running and taking up the processing power. Amazon itself estimates its

performance to be 26 EC2 Compute Units [23]. One EC2 Compute Unit is equivalent to a 1.0-1.2 GHz 2007 Opteron or Xeon processor [25].

Another instance that might fit out profile is Amazon's 'High I/O Quadruple Extra Large Instance' (hi1.4xlarge) which has a Xeon E5620 CPU running at 2.4 GHz and it has also eight cores but it has hyper threading enabled thus making it 16 cores. Amazon estimates its performance to be 35 EC2 Compute Units [23]. This instance is also on the Ubuntu 12.04.1 LTS.

As mentioned before under State of the Art Amazon uses Xen's paravirtualization.

4.2. Matching Instances to the server

In this section we are running the Phoronix Test Suite's Primesieve test on both Amazon instances and on our server to see how much a single computing unit differs from another. Figure 10 represents the gathered data. Our server falls perfectly between two Amazon's instances and as Amazon does not provide any more instance types between 26 and 35 EC2 Compute Units then this is the closest estimation we can make.

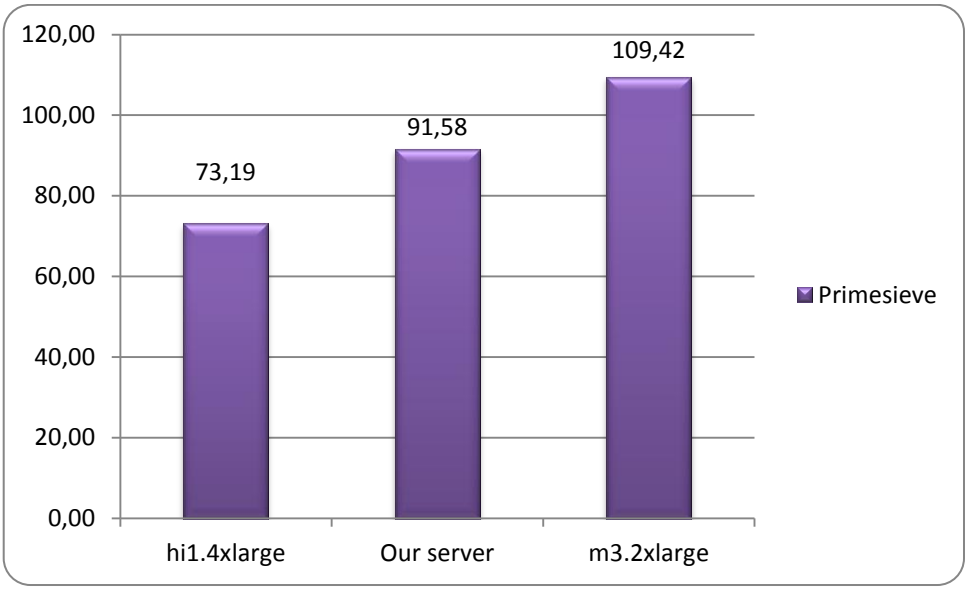

Figure 10. PTS Primesieve benchmark. On one instance. Units are in seconds. Smaller is better.

Next, we will see the results from the NPB Integer Sort benchmark. Figure 11 shows that when scaling from one virtual machine to two virtual machines our server performs very slow compared to the Amazon instances. This is because this test requires significant amount of data to be transferred between guests. Amazon instances were deployed into the same availability zone for both instance types and as Amazon has very hi-speed connections throughout their availability zones, their instances perform extremely well. M3.2xlarge

instance has a 1 Gbit network interface and hi1.4xlarge has a 10 Gbit network interface and this is clearly represented on the chart.

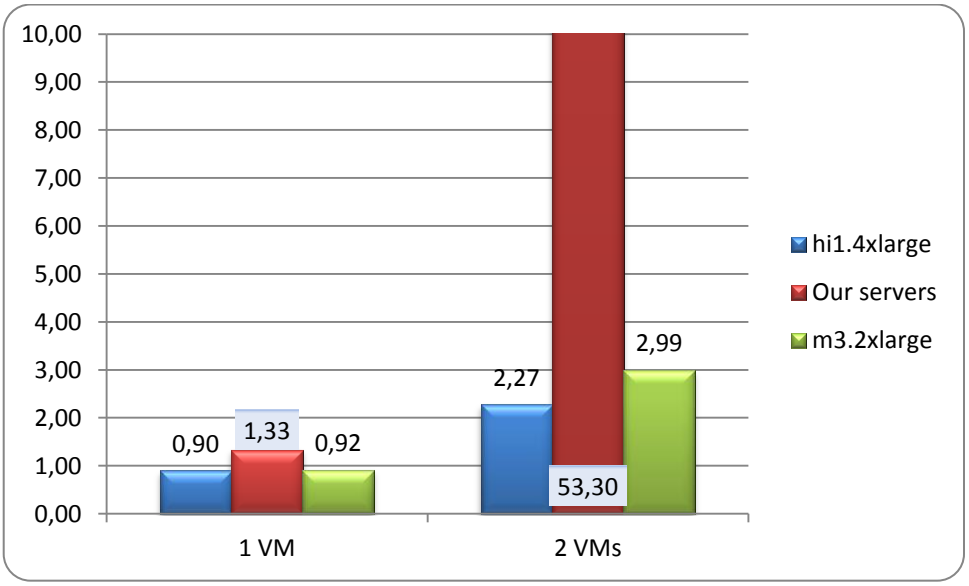

Figure 11. NPB Integer Sort benchmark. Units are in seconds. Smaller is better.

Building a fast network is very expensive and if not yet present then upgrading to a faster network might not be an option. Considering all of the previous information it might be cheaper money wise and faster performance wise to deploy scientific calculation on Amazon's cloud. Setting up an Amazon cloud is also a lot faster but as this cluster already exists then when not dealing with calculations which require heavy communication, then the University of Tartu can sufficiently use these servers.

Conclusion

All in all it seems that adding a virtualization layer alone does not affect the computing power very much but a slight delay (13-16% overhead in time) in disk operations should be taken into account when dealing with a job that requires I/O features.

When trying to scale the number of virtual machines on one physical machine performance tends to drop, so this is not a valid technique to improve performance for scientific computing purposes. Also a very dramatic performance loss for disk operations should be noted, up to 300% of performance loss. Communication between virtual machines in one host seems to take a very long time and Xen is heavily troubled by this defect.

When trying to scale the number of physical machines and keeping the number of virtual ones the same, performance of course improves to the point where the deployed problem will no longer benefit from more computing power but rather has to deal with transportation of data and splitting the problem into a larger number of pieces. Using four physical machines to host four virtual machines and comparing those results to the four physical machines in a cluster i.e. only a virtualization layer is added, results confirm that adding a virtualization layer does not affect computing power but disk I/O operations take 100% or more time to complete. Here, KVM handled networking very similarly to the host but Xen again had some overhead resulting the job to take double time.

KVM is currently a much better option than Xen, at least using these virtualization techniques described above.

All in all 144 tests were run to gather the data used for the analysis. All tests were run three times to provide a more accurate representation and to eliminate outliers.

Tests run in Amazon reveal their superiority over university in networking speed but when not using calculation which require heavy networking, the use of university's servers are plausible.

In the future we could look into the Xen anomaly and unexplainable rise in the performance in the NPB Block Tri-diagonal Input Output benchmark and in the Integer Sort benchmark.

Teaduslikus arvutusprotsessis riistvara virtualiseerimise hind

Bakalaureusetöö (6 EAP)

Allan Trukits

Resümee

Selle töö eesmärk on uurida riistvara virtualiseerimise negatiivseid aspekte, kui sooritatakse teaduslikke arvutusprotsesse, mis nõuavad suurt arvutusjõudlust. Tihti on nii, et teaduslikud probleemid on nii mahukad, et tulemuse arvutamiseks peab selle probleemi osadeks jaotama ja mitmetel arvutitel samaaegselt jooksutama.

Virtualiseerimine annab mitmeid eeliseid nagu seadistamise lihtsus, riistvara ja tarkvara lahtisidestus, väga kiire paigaldus ja konfiguratsiooni muutus ning elastsus. Kuid lisa virtualisatsioonikiht võib endaga kaasa tuua mitmeid puuduseid, eriti ressursi nõudlikele teaduslikele algoritmidele, mis rakendavad paralleelarvutus tehnoloogiaid.

Esimesena on välja toodud eelnevalt tehtud uurimustööd, mis on suuremal või vähemal määral analoogilised selle tööga, kuid need uurimustööd on tehtud mitu aastat tagasi ja vahepeal on kasutatud tarkvara edasi arendatud ning nendest on välja antud uuemad versioonid, mis annab alust arvata, et ka nendes eelnevates uurimustöödes saadud tulemused ei vasta enam tegelikkusele.

Teiseks on selles töös kirjeldatud kasutatud riist- ja tarkvara ning kirjeldatud iga jõudlustesti iseärasusi ja miks iga test on valitud. Testid jooksutatati Ubuntu operatsiooni süsteemil kasutades Xen ja KVM tarkvara virtualiseerimiseks. Testimiseks kasutatakse NASA poolt välja töötatud spetsiaalset tarkvara paralleelsete süsteemide jõudlustestimiseks – „NAS Parallel Benchmarking“. Samas kasutatakse ka Phoronixi poolt välja töötatud jõudlusteste testimaks üksikute arvutusinstantside jõudlust, kuna NPB on loodud just paralleelsüsteeme testima, siis ta ei pruugi anda adekvaatset hinnangut üksik instantside jõudlusele.

Jõudlustestidega mõõdetakse võrgulatentsuse, sisend-väljundite kiiruse ja protsessori jõudluse muutumist erinevates keskkondades ning mõju mälukasutusele.

Kolmandaks võrreldakse tulemusi ühe, kahe ja nelja arvuti suuruses pilves koos virtualiseerimisega ja ilma virtualiseerimiseta ning ühe füüsilise masina peal ühte, kahte ja nelja virtuaalmasinat.

Hüpotees on, et Xen ja KVM on jõudluse osas võrdsemad kui minevikus. Eelnevates uurimustöödest on selgunud, et Xen on edukamalt jõudlustestidega hakkama saanud kui

KVM. Lisaks võib arvata, et virtualiseerimine mõjub jõudlusele halvasti, kuna kasutusel on lisaks põhioperatsiooni süsteemile veel virtualiseerimistarkvara ja lisa operatsiooni süsteem. Samas võib võrgulatentsus ühes füüsilises masinas virtualiseeritud süsteemide pilves väheneda kuna võrgumeediumid ei ole enam piiranguks.

Tulemused aga näitavad, et virtualiseerimiskihi lisamine ei avalda erilist mõju protsessori ja mälu jõudlusele, küll aga mõjutab see oluliselt kõvakettale kirjutamise kiirust, isegi kuni saja protsendilist jõudlusekadu võib oodata. Suurendades virtuaalmasinate arvu ühel füüsilisel masinal jõudlus kahaneb nagu oli ka oodata, nii et see ei ole hea kasutusviis teaduslike arvutuste tegemiseks. Lisaks suurenesid siis oluliselt kõvakettale kirjutamise ja lugemise operatsioonide täitmisaeg ja ka suhtlus virtuaalmasinate vahel võrgus muutus väga aeglaseks. Teisalt kui suurendada füüsiliste masinate arvu jättes virtuaalmasinate arvu samaks, võib oodata jõudluse kasvamist nagu eeldatud. Füüsiliste masinate arvu suurendamist saab muidugi jätkata kuni tuleb ette antud ülesande piir, kust maalt ei ole mõistlik seda enam rohkemateks osadeks jaotada, sest rohkem aega kulub võrgulatentsutele ja probleemi osadeks jaotamisele.

Väga üllatav on tulemus, et KVM oli testides oluliselt edukam kui Xen, otsest põhjust sellele on raske tuua, aga testid räägivad enda eest.

Kokkuvõttes sooritati 144 testi, et antud andmeid analüüsi tegemiseks koguda. Kõiki teste sooritati kolm korda, et eemaldada üksikute halbade kokkusattumuste tõttu piirtulemusi ja et anda täpsem hinnang testitulemustele. Järeldus on, et virtualiseerimist võib kasutada teaduslikeks arvutusteks ilma olulisi jõudlus kadusid märkamata, küll aga peab arvestama kõvakettale kirjutamise kiiruse kahanemises. Virtualiseerida tuleks aga nii, et on mitu füüsilist arvutit, kus samal ajal on ühe ülesande jaoks virtualiseeritud füüsilised masinad üks-ühele. Samas võib muidugi olla rohkem virtuaalseid masinaid jooksmas, aga neid ole ole mõistlik samal ajal arvutusteks kasutada. Selline virtualiseerimine tagab selle, et on tagatud turvalisus ja eraldatus erinevatele inimestele antud ressursside osas ja samas ei kaotata oluliselt jõudluse arvelt, kui on teada, kunas arvutid reaalselt kasutuses on.

Appendixes

Appendix #1:

Xen configuration:

```
dir = /mnt/xen
install-method = debootstrap
size = 5Gb
memory = 1024Mb
swap = 128Mb
fs = ext3
dist = precise
image  = sparse
dhcp = 1
bridge = br0
kernel = /boot/vmlinuz-`uname -r`
initrd = /boot/initrd.img-`uname -r`
arch = amd64
mirror = http://archive.ubuntu.com/ubuntu/
ext3_options    = noatime,nodiratime,errors=remount-ro
ext2_options    = noatime,nodiratime,errors=remount-ro
xfs_options     = defaults
reiserfs_options = defaults
btrfs_options   = defaults
boot = 1
passwd = 1
serial_device = hvc0
disk_device = xvda
maxvcpus = 4
vcpus = 4
```

References

[1] Todd Deshane, Zachary Shepherd, Jeanna N. Matthews, Muli Ben-Yehuda, Amit Shah, Balaji Rao. Quantitative Comparison of Xen and KVM, http://140.110.240.196/grid/raw-attachment/wiki/KVM_vs_Xen/Quantitative%20Comparison%20of%20Xen%20and%20KVM.pdf, 2008. Online. URL last visited on 5th of May, 2013.

[2] Andrea Chierici, Riccardo Veraldi. A Quantitative Comparison Between Xen and KVM, http://iopscience.iop.org/1742-6596/219/4/042005/pdf/1742-6596_219_4_042005.pdf, 2010. Online. URL last visited on 5th of May, 2013.

[3] Lucas Nussbaum, Olivier Mornard, Fabienne Anhalt, Jean-Patrick Gelas. Linux-based Virtualization for HPC Clusters, http://hal.inria.fr/docs/00/42/56/08/PDF/linux-virtualization-mls09.pdf, 2009. Online. URL last visited on 5th of May, 2013.

[4] Xen, http://en.wikipedia.org/wiki/Xen. Online. URL last visited on 5th of May, 2013.

[5] Kernel-based Virtual Machine, http://en.wikipedia.org/wiki/Kernel-based_Virtual_Machine. Online. URL last visited on 5th of May, 2013.

[6] Virtualization, http://en.wikipedia.org/wiki/Virtualization. Online. URL last visited on 5th of May, 2013.

[7] NAS Parallel Benchmarks, http://www.nas.nasa.gov/publications/npb.html. Online. URL last visited on 5th of May, 2013.

[8] Message Passing Interface, http://en.wikipedia.org/wiki/Message_Passing_Interface. Online. URL last visited on 5th of May, 2013.

[9] MPICH, http://www.mpich.org/. Online. URL last visited on 5th of May, 2013.

[10] Gfortran, http://gcc.gnu.org/wiki/GFortran. Online. URL last visited on 5th of May, 2013.

[11] The Phoronix Test Suite, http://www.phoronix-test-suite.com/. Online. URL last visited on 5th of May, 2013.

[12] Subhash Saini, David H. Bailey. NAS Parallel Benchmark (Version 1.0) Results 11-96, http://www.nas.nasa.gov/assets/pdf/techreports/1996/nas-96-018.pdf, 1996. Online. URL last visited on 5th of May, 2013.

[13] Problem Sizes and Parameters in NAS Parallel Benchmarks, http://www.nas.nasa.gov/publications/npb_problem_sizes.html. Online. URL last visited on 5th of May, 2013.

[14] William Saphi, Rob Van der Wijngaart, Alex Woo, Maurice Yarrow. New Implementations and Results for the NAS Parallel Benchmarks 2, http://www.nas.nasa.gov/assets/pdf/techreports/1994/npb_2.2.pdf. Online. URL last visited on 5th of May, 2013.

[15] Parkson Wong, Rob F. Van der Wijngaart. NAS Parallel Benchmarks I/O Version 2.4, http://www.nas.nasa.gov/assets/pdf/techreports/2003/nas-03-002.pdf, 2003. Online. URL last visited on 5th of May, 2013.

[16] Mobile & Cloud Computing Laboratory, http://mc.cs.ut.ee/. Online. URL last visited on 5th of May, 2013.

[17] MPI: Multicore Host Files, http://cs.calvin.edu/curriculum/cs/374/homework/MPI/01/multicoreHostFiles.html. Online. URL last visited on 5th of May, 2013.

[18] Satish Narayana Srirama, Oleg Batrashev, Pelle Jakovits, Eero Vainikko. Scalability of parallel scientific applications on the cloud,

http://iospress.metapress.com/content/a3r8511831505325/. Online. URL last visited on 5th of May, 2013.

[19] Virtualization, http://en.wikipedia.org/wiki/Virtualization. Online. URL last visited on 5th of May, 2013.

[20] Amazon EC2, http://aws.amazon.com/ec2/. Online. URL last visited on 5th of May, 2013.

[21] OpenStack, http://en.wikipedia.org/wiki/OpenStack. Online. URL last visited on 5th of May, 2013.

[22] Huan Liu. Amazon's Physical Hardware and EC2 Compute Unit, http://huanliu.wordpress.com/2010/06/14/amazons-physical-hardware-and-ec2-compute-unit/. Online. URL last visited on 5th of May, 2013.

[23] Amazon instance types. http://aws.amazon.com/ec2/instance-types/. Online. URL last visited on 5th of May, 2013.

[24] Counting Bytes and FLOPS, http://www.cs.virginia.edu/stream/ref.html. Online. URL last visited on 5th of May, 2013.

[25] Amazon EC2 Compute Units, http://aws.amazon.com/ec2/faqs/. Online. URL last visited on 5th of May, 2013.

Non-exclusive licence to reproduce thesis and make thesis public

I, Allan Trukits
(date of birth: 13.12.1988),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright, The Cost of Virtualization for Scientific Computing, supervised by Pelle Jakovits,

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **10.12.2013**