Virtualisation for Grid-Computing

Marcus Hardt
Institute for Scientific Computing
Hermann-von-Helmholtz-Platz 1
76344 Eggenstein-Leopoldshafen
Germany
hardt@iwr.fzk.de

Abstract

One problem in distributed computing is bringing together application developers and resource providers to ensure that applications work well on the resources provided. A layer of abstraction between resources and applications provides new possibilities in designing Grid solutions.

This paper compares different virtualisation environments, among which are Xen (developed at the University of Cambridge / UK) and User-Mode-Linux (a linux community effort). The differences in architecture and features will be presented. The results of our intensive performance measurements that have been carried out on all of these virtualisation environments will be discussed.

Furthermore use cases that highlight solutions to typical problems in distributed computing with particular emphasis on Grid computing will be presented.

1 Introduction

Deployment of large-scale distributed applications in todays grid installatons is a very difficult business. This is due to the large amount of parameters required to fully describe a computing system. These parameters include the architecture, the choice of operating system, the chosen OS distribution and the version being used. Even the compile-time settings used for creating certain libraries might be required to describe a system. In some applications it was found that even different settings for the compilation of the mathematical library `libm` can cause changes in the final results of calculations. In consequence high energy physics (HEP) experiments, such as D0 and CDF, have prepared certification toolkits which have to be run on new resources before they can be added to the experiments' resource pool. These certification toolkits cover the major problems that experiment software typically encounters, including to ensure that the experiment software will install properly on the remote machine.

The easiest solution for this problem would be to install well defined software on well defined hardware for all resources in the Grid. This contradicts one of the key design principles of the Grid, to support heterogeneous environments and is obviously not realistic. Furthermore this is unfeasible in a fast-moving, global environment with dynamics due to ongoing development. In the European projects Data Grid (EDG) and CrossGrid (CG) this approach was taken as far as the installed software is concerned, because the main focus of both projects was the development of middleware and early adaptors' application software. However, this tied a considerable amount of manpower to system administration tasks.

Another issue are the conflicts between system administrators, middleware and application developers. All use their own systems to develop their applications. This means that all own the systems they develop on, thus being able to tweak them to their needs. This results in different, potentially incompatible, configurations, resulting in incompatible assumptions being made on the resources where the software will be used. In a Grid environment their applications are supposed to work on all kinds of resources administered by a multitude of different administrators with typically different policies and restrictions.

In other words, there is currently no common understanding of how each node for distributed computing should be designed. A layer of abstraction, which gives each party enough freedom, might solve this problem. This paper will present an overview about different virtualisation environments and show a benchmark comparison.

2 Virtualisation Approaches

2.1 Z-Series

Virtualisation is most renown from IBM z-Series hardware. This solution depends on the specific z-Series hardware together with the specialised operating system. Although many virtual linux systems can be booted on this system, this kind of virtualisation is not referred to in this paper.

2.2 User Mode Linux

User Mode Linux (UML) is an opensource solution for linux-on-linux virtualisation. UML is essentially a kernel port from the default x86-hardware to linux as a hardware. The UML kernel is an executable just like `ls`. It can be run by any user. Parameters specify the disk image to use or the IP address to assign. Although UML is a userland tool it must be said that networking is only available after interaction with administrative privileges in order to set up the required network interfaces.

The virtualisation takes place at the device driver level. For most of the native linux devices UML provides a virtual device that can be used within the Virtual machine. This can be thought of as passing on the native devices to the virtual machines.

In order to support clustering a large number of similar machines, UML supports a copy on write (cow) filesystem. It consists of one master image file plus one per UML instance that contains the changes against the master image on a block level. Most probably newer will make use of the transparent overlay filesystems, recently supported by the linux kernel.

2.3 Xen

Xen [4] approaches to bring Virtualisation, as known from Mainframes, to the Intel x86 Architecture. Other than many known virtualisation systems the Xen Team did not aim for full virtualisation. Instead, the approach of "*paravirtualisation*" was created.

While only a very limited amount of privileged calls has to be virtualised in order to run more than one operating system concurrently, paravirtualisation implies modifications of the guest operating systems. Such modifications are usually limited to just a few thousand lines of code, compared to the approximately 6 million lines of code of the Linux kernel and thus do not represent a major change. Not only a large amount of open operating systems was ported to run on Xen. Also windows XP was modified by MS-Research in order to run on Xen, but was never released as a product.

One design goal of Xen was to keep the application binary interface (ABI) unchanged. This means that application binaries will run inside virtual machines without any modifications. For architectural details see [1, 2].

The host system is usually referred to as Domain-0 or dom0. This is a domain – or virtual machine – with the special privilege of communicating with the new underlying Xen layer. Domain-0 is also in charge of accessing the hardware and providing virtualised interfaces to the guest domains, which are called domUs. The domUs are the actual *virtual machines*. On todays computers their number is mainly limited by the available resources, most often the amount of RAM.

A set of userspace tools is provided for controlling the virtual machines. This comprises the basic functions, like creating and destroying virtual machines, connecting consoles to them as well as more advanced functions for stopping/starting, saving and even migrating virtual machines between different physical hosts. The filesystem for guest systems can be placed into raw partitions, on a Logical Volume Manager (LVM) partition or simply inside a file. They are made available via the virtual block device (VBD) driver to the guest domain. Booting a system via NFS-root is possible as well.

2.3.1 Features

Xen can stop, resume and kill virtual machines, which implements the various funcions of a PC's powerbutton.

Furthermore, Xen allows to migrate domains to different machines (on the same subnet while they are running. Network connections *stay active* during the migration. In the case of "Live migration", no noticeable down-time can be observed. First, all domain-specific data is copied. In the next step, only changes between the "pre-copying" stage and the actual migration are sent to the new host.

Unfortunately, we were only able to use migration in small test installations. This is because the image file needs to reside on shared network storage, so it does not have to be migrated. Using NFS to share the images between hosts was not possible because of a known issue of the Linux kernel when using loop-back files over NFS. The Xen team suggests using GNBD (Network Block Device) or i-SCSI (SCSI over IP). Root-mounted NFS may as well be an option.

2.3.2 Issues

One issue of current Xen versions is the incompatibility with the TLS implementation of most glibc distributions. While Xen can use a slower compatibility mode if this implementation is present, disabling TLS is suggested. Although this is easily done by removing `/lib/tls` problems occur during system updates which will occasionally bring back the tls libraries in updated versions. It must be said that a xen-compatible version of these libraries is in development.

Another issue is the disk-i/o performance delivered by Xen. Older measurements observed, that some Xen versions showed a very fast throughput, when image-backed rather than partition-backed. Newer Xen versions do not perform well on image-backed installations. The Xen Team suggests using LVM or partitions instead.

3 Commercial Product

The license of the commercial product we evaluated does not allow to publish performance measurements. Since we wanted to publish them we chose to keep the product name secret.

The approach is totally different to the above. This system is the only one that provides real virtualisation. Its own MMU intercepts accesses to memory and redirects them accordingly. Virtual devices like ethernet or harddrive and sound can be connected. In the case of sound and graphics special drivers are required and provided.

4 Virtualisation Use Cases

4.1 gLite Installation course

As a part of the courses held at GridKa School 2005 [**?**] a gLite intallation course was planned. It was required to train up to 40 students in small groups of 3 people. The hardware constraints of glite required 5 PCs per group plus a set of 5 machines for central services. In this sums up to 70 machines required for such a course. Experience from a similar event that took place in the previous year showed that this is generally not desirable – especially due to the fact that physical access to the machines will be required in order to install the basic operating system. For GridKa School 2004 this involved moving 70 PCs from the compute-center into the office building and connecting all PCs to network, power, keyboards and screens. – Obviously not a perfect solution.

Essentially due to the lack of availalbe computers Xen was investigated and proved to be a considerably good solution due to several facts:

4.1.1 Direct access to virtual machines

Students can access the virtual machines from the host machines as if they are directly connected to a terminal. I.e. Basic OS installation or network configuration can be done without special cabling (i.e. serial consoles, console servers or equivalent).

4.1.2 Consolidation of machines

The goal of an installation course is to understand how to install the software rather than providing high performance computing environments. Thus we integrated all the five machines that required per group onto one physical machine. For this task host machines with only P-III 700Mhz and 1GB RAM were entirely sufficient.

4.2 IT consolidation

A current trend in IT divisions is to use virtualisation in order to consolidate services. Following this idea we have virtualised our Grid environment.

Part of our duties within the EGEE project is to run trainings on grid middleware, in particular installation courses. Very unfortunate for this type of course is the fact that the EDG- based Grid middleware LCG-2.4 is designed to scale from large to very large installations. Thus the installation assumes one dedicated computer for each task, as described above.

Although it is in principle possible to manually integrate several nodes onto one, this is not foreseen by the installation procedures and thus requires a good inside knowledge of the LCG software. Using Xen we installed one node running three virtual machines, running the required central services. For every group of students one node was set up to run five virtual machines, each of which has to be configured to run one of the LCG services as a course goal.

The hardware available for the courses were 16 dual Xeon-700s with 1GB of RAM and 40GB harddrives. The main constraint on the hardware is the memory, because Xen does not support memory over-subscription. Accordingly we chose to create swap files and pass them to Xen. Together with the image file, that contains the whole file system for use inside the virtual machine, the installation is complete and requires less than 8GB of disk space.

We are currently working on using a very similar setup in order to run a gLite (EDG-middleware successor) installation course. One problem here is that the students will need physical access to the machines they are installing. The main advantage in this scenario is that by using the virtualisation environment, we can avoid to physically transport the required 66 computers to the course location, because it gives “virtual physical access” to the machine, i.e. full control over the virtual machines is available from inside the physical host system, to which remote logins are enabled already.

4.3 Using Windows Desktops

In our research center roughly 4000 PC are installed with Windows for doing office kind of working. Summing up office hours with respect to weekends and holidays turns up that these PCs are only operated one third of their time, thus two third of their time they could be added to the local cluster. Unfortunately none of the windows OS flavours supports the LHC grid middleware. However, virtualisation tools exist that can be used to run a linux PC on top of windows. The performance loss is at around 10% (as we will show below), thus this appears feasible to do.

4.4 Whole image job submission

The work done on the installation course setup led us to the conclusion that it should be possible to use this technology in order to solve a more general problem in LCG based – and probably many other – grid environments. We found already, that it is possible to run all infrastructure services inside virtual machines. This usecase works, if the workernodes support Xen, but are not running inside a virtual machine themselves.

This would allow application developers to use a virtual machine inside which he develops his application. This virtual machine can run any Linux distribution and can be as customised and tuned as required by the developer. The underlying image from which the virtual machine was booted can be sent to any Grid node, where it is booted. The developer must take care that processing is started appropriately, that the results are stored in a suitable place so that access to it is ensured, once the job terminated. The developer can utilise his favourite grid middleware to accomplish this.

The overhead introduced by sending around images containing a whole Linux plus Grid middleware plus application software distribution is less than one would expect at first sight. This has several reasons. First of all, replica services are widely deployed in their second and third generation. They allow to register a set of images to the Grid. Resource brokerage systems are available that can decide to start the computational jobs at a site where an appropriate image is provided. Secondly, developers are expect to use existing images and update the bits of software they developed inside that image rather than creating a new image for every test they need to run. The main advantage is still that they only develop on a homogeneous system.

4.5 Cluster Load Management

One more benefit of virtualisation using Xen is that it allows to migrate images inside a cluster.

Today in HEP related environments, clusters are used in the way that incoming jobs are sent to one workernode by the batch system. The job stays there until it is finished. Usually SMP machines are used as workernodes, resulting in the usual load of two jobs per node. Typically there are different jobs or different phases of jobs. These phases can be differentiated into i/o intensive phases and CPU intensive jobs.

When combining the migration functionality with appropriate monitoring, a flexible load management functionality can be implemented. It would be designed to ensure that complementary jobs (i/o bound vs. CPU bound) jobs share one node. It is predictable that in this way the overall efficiency of the cluster can be easily improved.

5 Performance

Most Performance measurements took place on the same hardware, a Sun Fire V20 with 2GHz Opteron, 80GB SCSI harddisk and 4GB of RAM. Only the User Mode Linux (UML) benchmarks took place on an older P-III 700MHz with 1GB of RAM and IDE disk. The performance measures shown for UML are the relative metrics to the plain measurements on the P-III. All other measurements are relative to the Opteron.

In order to get a feeling for the different subsystems of the hardware we ran a set of different benchmarks that stress the different hardware subsystems.

All benchmarks were run 1-16 times at the same time on the same hardware. For the virtual machines this means that all the virtual machines were running the same benchmark at the same time. Especially for more than 5 virtual machines running, there was a considerable effect that we called load-inbalance. It results in some VMs finishing 4 benchmarks while others only finish 2 or don't even finish at all. This happens although we confirmed from the commandline, that the benchmarks actually ran.

The commercial product did not succeed to easily run many parallel

5.1 CPU

The cpu benchmark (fig. 1) chosen was the *four in a row* benchmark from *freebench.org*. This is an integer-bound cpu-intensive benchmark and it is free software. The CPU2000 benchmark was not run because its memory requirements could not be met, when running 16 VMs on a 4GB RAM machine. It can be seen that we used a dual CPU machine, because the performance on 2 parallel runs is as fast as on one. Almost all virtualisation environments run at the same speed as technically possible, indicated by the Opteron-SMP measurement.

5.2 Memory

The memory benchmark (fig. 2) does not reveil a large performance-difference than the CPU benchmark does. We used pcompress2, from the same *freebench.org*.

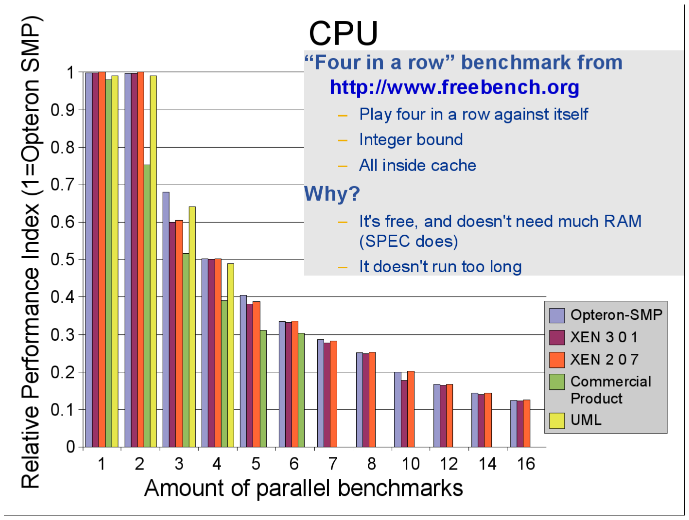

Figure 1. The Since UML does not scale well enough, we did not run 8 parallel VMs on it.

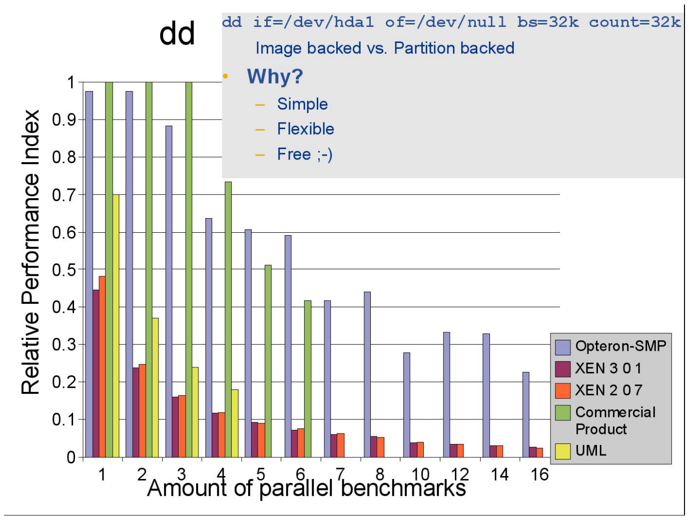

Figure 3. The free unix tool dd was used to read 1GB of data from the disk.

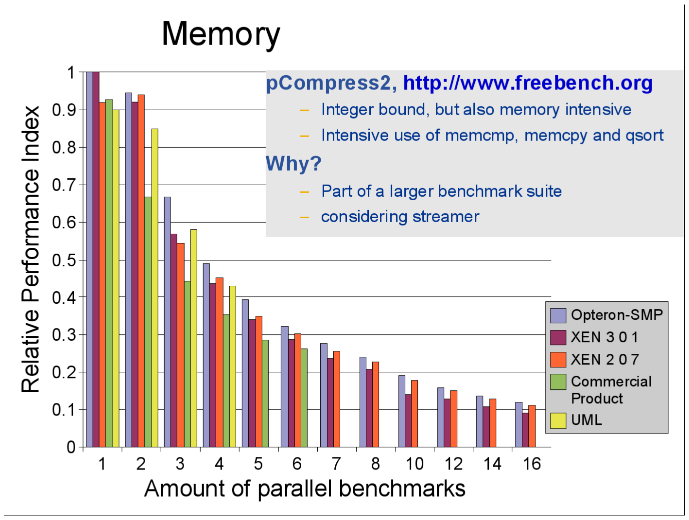

Figure 2. The memory benchmark pcompress2 makes use of memcopy and qsort

5.3 Disk

For the disk benchmark (fig. 3) we read 1GB from the harddrive of every virtual machine. This size allows for caching benefits by the underlying virtualisation environment. This effect does not take place for Xen, partially for UML whereas the plain Opteron and the comercial system do profit from caching.

5.4 Kernel

The kernel benchmark (fig. 4) was run with "*-j4*" in order to take effect of the two CPUs in the plain machine. We can see that none of the virtualisation environments took advantage of the many cpus. Newer versions of Xen as well as the commercial product can, however, we didn't use this feature for the sake of comparability. With two and more parallel runs we see that the commercial product and UML scale less than Xen does.

6 Conclusion

Virtualisation provides an effective method for consolidating hardware in Grid development and Grid training environments. It has proven to work in our development environment for installation courses.

Xen provides a stable and free, yet performant environment for virtualisation. We do not foresee critical problems in its application to production Grid environments, especially since the Xen team has a strong focus on security and is actively developing Xen further. However, a minor issue are incompatible versions of the TLS libraries that Xen is currently incompatible with.

The Commercial product seems to scale better performancewise, but unfortunately suffers from some load inequality when highly loaded.

Virtualisation appears to be mature enough to provide a solution for the problems that arise with the need for different environments or Linux distributions that are required when providing resources to the Grid.

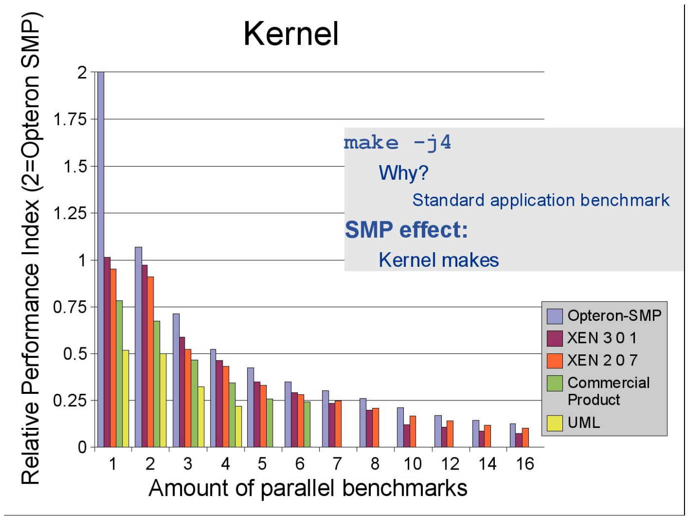

Figure 4. The kernel compilation on an SMP system shows less than 5% performance loss for the Xen system. User Mode Linux shows about 50% loss. Xen proves to scale very well to 16 virtual machines.

References

[1] Ian Pratt. Xen and the Art of Virtualization. Technical report, 2003. URL http://www.cl.cam.ac.uk/netos/papers/2003-xensosp.pdf.

[2] Ian Pratt. Xen 3.0 Virtualisation, 2005. URL http://makeashorterlink.com/?A25212A9B.

[3] T. Neward. Self modifying websites, 2002. URL http://makeashorterlink.com/?U20A1269B.

[4] The Xen Team. *Xen project homepage*, 2005. URL http://xen.sf.net.