A single-computer Grid gateway using virtual machines

Stephen Childs, Brian Coghlan, David O'Callaghan, Geoff Quigley, John Walsh
Department of Computer Science
Trinity College Dublin, Ireland
Firstname.Lastname@cs.tcd.ie

Abstract

Grid middleware is enabling resource sharing between computing centres across the world and sites with existing clusters are eager to connect to the Grid using middleware such as that developed by the LHC Computing Grid (LCG) project. However; the hardware requirements for access to the Grid remain high: a standard LCG Grid gateway requires four separate servers. We propose the use of Virtual Machine (VM) technology to run multiple OS instances, allowing a full Grid gateway to be hosted on a single computer. This would significantly reduce the hardware, installation and management commitments required of a site that wants to connect to the Grid. In this paper, we outline the architecture of a single-computer Grid gateway. We evaluate implementations of this architecture using two popular open-source VMs: Xen and User-Mode Linux (UML). Our results show that Xen outperforms UML for installation tasks and standard gateway operations. Configuration is similar to that of sites running multi-computer gateways, making it easy to keep site installation profiles synchronised. Our VM gateway architecture provides a low-cost entry path to the Grid and will be of interest to many institutions wishing to connect their existing facilities.

1 Introduction

1.1 Context

The Grid-Ireland project provides grid services above the Irish research network, allowing researchers to share computing and storage resources using a common interface. It also provides for extra-national collaborations by linking Irish sites into the international computing grid. The national infrastructure is based on middleware from the LHC Computing Grid (LCG) project [10]. LCG provides a common software distribution and site configuration to ensure interoperability between widely distributed sites. The current release (2.2.0) uses LCFGng [1] for network installation of nodes according to configuration profiles stored on an install server.

Grid-Ireland currently comprises an Operations Centre based at Trinity College Dublin (TCD) and six nationwide sites. The Operations Centre provides top-level services (resource broker, replica management, virtual organisation management, etc.) to all sites. Each site hosts a Grid access gateway and a number of worker nodes that provide compute resources. We aim to make Grid services accessible to a far higher proportion of Irish research institutions in the near future. To achieve this goal we must ensure that the hardware and personnel costs necessary to connect a new site to the Grid are not prohibitive.

A standard LCG gateway configuration makes significant hardware demands of a site. A minimum of four dedicated machines are normally required: an install server, providing a configuration and software repository for all nodes; a computing element (CE), providing scheduled access to compute nodes; a storage element (SE), providing data management, and a user interface (UI) providing for job submissions from users.

We propose that it is possible to reduce the hardware commitment needed by running all gateway services on a single physical machine. This machine would host a number of VMs acting as logical servers, with each VM running its own OS instance to maintain isolation between servers. As each VM would appear to be a real machine (both to the server software and to users), the need for special configuration relative to the existing gateways would be removed.

1.2 Aims

We aim to reduce hardware, personnel and space costs per site: the overhead of installing and maintaining four or more server machines is excessive for small sites that wish to explore Grid functionality. Smaller sites may only have a few users and cluster nodes initially and a single server solution would be much more acceptable than the standard configuration. We also want to limit the divergence from a standard Grid site configuration so we can use basically the

Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA'05)
1550-445X/05 $20.00 © 2005 IEEE

same configuration data (LCFG profiles, software package lists) for both multi-machine sites and single-machine sites. We need to provide for central management of remote sites. We currently manage gateway servers at remote sites using console redirection and Secure Shell (`ssh`) access; any new setup must provide the same level of access. Finally we must provide a simple installation process that can be performed remotely.

1.3 Benefits of VM approach

We propose to run multiple logical servers in separate VMs on a single physical machine. The main alternatives to this are: i) to dedicate a physical machine to each server, and ii) to run all services as normal processes within a single OS instance. We now briefly list the advantages of our proposed approach over these alternatives.

Relative to a multi-box solution there is obviously a lower hardware cost: one server costs less than four servers! There is also easier management: a single machine solution makes fewer demands on site administrators in terms of machine room space, power requirements, network connections, etc. There are also advantages relative to a single-OS solution. Mainstream OSes provide relatively weak isolation between user-level processes, making it easy for misbehaving applications to monopolise system resources. A good VM monitor (VMM) will provide resource control facilities that allow administrators to set hard limits on the amount of resources available to any one VM. Another benefit is that smaller sites using VM solutions to run multiple OS instances on a single machine can use the same basic configuration as larger sites with multi-machine gateways.

1.4 Outline

In the remainder of this paper we describe our approach to testing the feasibility of building Grid gateways using VMs. In Section 2 we discuss the factors that will determine our choice of VMM, in Section 3 we describe the architecture of gateways built on two VMMs, and in Section 4 we present performance measurements for both platforms. In Section 5 we make some observations based on our experience of deploying VM technology. Section 6 discusses related work, and finally Section 7 summarises our findings.

2 Choosing a VMM

Making a good choice of VMM is crucial to building a secure, fast system that is easy to manage. In this section, we outline the technical and administrative requirements that the gateway software makes of a VMM. We also briefly describe a range of currently available VMMs, and choose representative VMMs for evaluation.

2.1 Requirements

We aim to run all gateway services quickly, securely and reliably on a single machine and so require a VMM to provide the following features:

Isolation: In a single-box solution, it is important for the VMM to provide isolation between VMs, so that even a catastrophic OS failure in one VM will not affect the others. (A hardware failure will inevitably affect all hosted OSes, but this risk could be mitigated by providing a backup machine to act as a failover.)

Storage: The logical servers each have different storage requirements but must share a limited set of local disks: the VMM should provide a flexible means of sharing the available disk space between hosted nodes to reduce the need for tricky repartitioning in the case of file systems filling up.

Resource control: The various servers also have different CPU requirements: the VMM should provide a means for controlling CPU utilisation. For example, to preserve interactive performance on the UI it may be necessary to throttle back the CPU utilisation of the other nodes. It would also be useful to be able to partition other resources such as disk and network bandwidth and physical memory.

*Low **overhead:*** The VMM should not impose a high performance overhead or significantly reduce system reliability. This is particularly an issue during I/O intensive operations such as installation/upgrade: while almost all VMMs can run compute-bound code without much of a performance hit, few can efficiently run code that makes intensive use of OS services. As the gateway will host the User Interface, the VMs must provide good interactive response times.

The VMM should also provide features to facilitate management of VM nodes. Such features typically include access to consoles for each VM, a facility for storing VM configurations, and tools for displaying and controlling VMs' resource usage.

2.2 Overview of VMMs

A virtual machine system provides a user with a complete OS environment tailored to his applications and isolated from other users of the computer. The virtual machines are controlled by a monitor (VMM), which enforces protection and provides communication channels. In the past, VM technology was most widely applied in mainframe computing, for example in IBM's VM/370 system [8]), where it was used to allow many users to share the resources of a single large computer.

Recently, interest has grown in implementing VMMs on commodity hardware and the past few years have seen a stream of commercial and open-source VMMs which provide varying levels of virtualisation. Full virtualisation vir-

tualises a complete instruction-set architecture: any OS that will run on the underlying hardware will run on the VM. Examples include VMWare [3], a commercial product that provides full x86 virtualisation on both Windows and Linux. Para-virtualisation presents a modified interface to guest OSes, which must be ported to the new VM "architecture". Xen **[2]** is a para-virtualised VMM which supports Linux and BSD-based guest OSes. Finally, system call virtualisation provides an application binary interface that enables guest OSes to run as user-space processes. User Mode Linux (UML) [4] is a port of the Linux kernel to run in user-space; it can be run on an unmodified host OS although kernel modifications are available that improve performance.

In our evaluation, we focus on Xen and UML. These are both open-source projects, allowing us to customise the code if we need to. They are also stable projects with active user communities to provide support. We have excluded commercial VMMs from our experiments due to cost considerations and licensing restrictions.

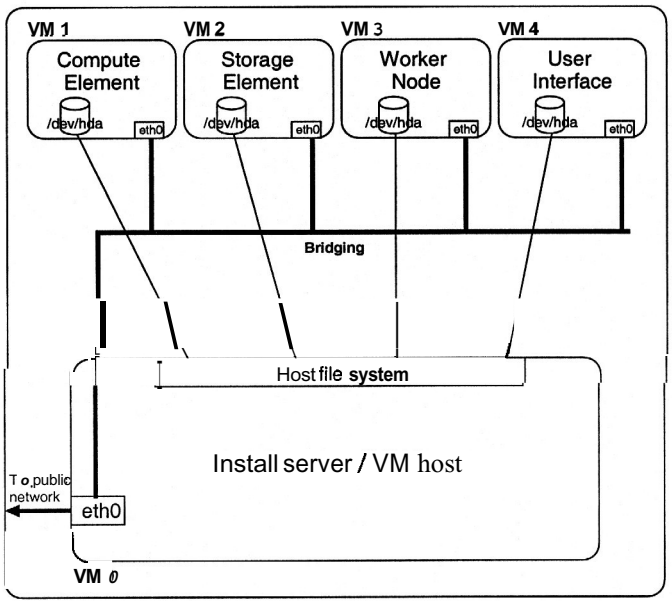

Figure 1. Architecture of grid gateway

3 Gateway architecture

The basic architecture is the same under both Xen and UML: the host OS runs an LCFGng [1] server, which is used to install a CE, an SE, a UI, and a Worker Node, each of which run in their own VM. (While it is not strictly necessary to provide a Worker Node, it is useful for testing the system before real cluster nodes have been integrated.) Figure 1 shows the high-level structure of the system.

3.1 Xen

VM setup The host OS runs in a Xen privileged VM that includes full device driver support for the machine's hardware. This VM exports devices to the VMs it hosts. Each VM is allocated 200 MB of memory and 20 GB of disk space.

Networking The Xen kernel creates virtual network interfaces for the VMs, and these virtual interfaces are bridged on to the real Ethernet address of the host machine (using standard Linux bridge utilities). Each VM has its own public IP address allowing full access from outside the gateway machine.

Storage The file systems for the VMs are backed by sparse files on the host file system. Linux loopback devices are attached to these files and then exported to the VMs using Xen's virtual block device system. Within the node's VM these devices appear as standard disks (e.g. `/dev/hda`) and so standard operations such as partitioning can be carried out as normal.

Configuration The configuration for each server node is stored on the VM host (install server) in a directory hierarchy under `/var/xen-grid`. Each node's directory contains one subdirectory (`config`) where configuration data (memory requirements, IP address, disk size) is stored, and another (`fs`) which holds the sparse file backing the file system.

The LCFG profile specifies which packages are installed on the node, along with site-specific information (network configuration, disk partitions, server locations, user accounts, etc.). No changes were needed to make these compatible with Xen.

Control of VMs We created a control script on the server for starting and stopping the VMs. The script uses the information stored in simple configuration files to generate commands for the Xen domain creation tools. In install mode, the script creates a sparse file and attaches it to a loopback device. The Xen domain is then booted from an NFS root file system on the LCFG server, with `init` set to point to the LCFG install script. This initiates the LCFG install process, which partitions the disk, retrieves the node profile from the install server and installs the OS and software packages specified in the profile. In standard boot mode, the script attaches a loopback device to the backing file, and then boots the Xen VM from the appropriate partition on the exported device.

Remote management Each gateway must support remote management so that Grid-Ireland staff can initiate and control upgrade procedures. Xen provides full console redirection for each of the server VMs. The VMs also run `ssh` daemons allowing direct access once network service on the VM are active.

3.2 UML

The structure of the UML-based gateway is similar to that of the Xen gateway. In the following description, we focus on the elements that are different.

VM setup As UML VMs are actually user-level processes, we run each instance as a separate user, providing a degree of isolation. The host runs a Linux kernel patched with the Single Kernel Address Space (SKAS) modifications to improve UML performance. Each VM is assigned 4 GB of disk space and 200 MB of memory.

Networking Networking is provided using the Linux TUN/TAP [9] driver, which creates virtual Ethernet devices for each of the VMs. These virtual devices are bridged onto the real Ethernet interface, and appear on the network as if they were real machines. This is equivalent to the Xen network configuration, with the only difference being that Xen creates its own virtual interfaces without needing to use the TUN/TAP driver.

Storage Each partition (root, boot and swap) is backed by a file on the host computer's file system under the hierarchy `/var/uml/`*node*`/fs`. On the UI, we also directly mount a disk partition on the host to provide extra space for /home.

Configuration Changes to the LCFG configuration were necessary for compatibility with UML. We modified the LCFG install script, disk partitioning code and node profiles to support the `devfs` file system used by UML. Configuration data for the whole system is stored separately to that specific to particular nodes.

Control of VMs As with Xen, we have developed an extra control script that automates the creation of file systems, configuration of networking and other tasks needed before VM execution can start. This script is first called at system startup to initialise the UML networking, and is also used to start and stop VMs, translating the configuration data into command-line parameters for UML.

Remote management We use the Linux `screen` utility to provide re-attachable console access to the VMs; `ssh` access is also provided.

4 Evaluation of Xen and UML gateways

In this section, we present the results of a performance evaluation of our Xen and UML gateways. We focus here on measuring the performance of Grid gateway applications; a detailed comparison of UML and Xen performance on standard benchmarks may be found in [2]. We ran all tests on the same machine: a Dell PowerEdge 1750 server with two 2.4 GHz processors amd two 140 GB SCSI disks. (This is the same configuration that will be deployed to the various sites nationwide.) Both UML and Xen used the same external network server for NTP and DNS.

4.1 LCFG installation

Our first test measures the speed of a first-phase LCFG installation. During this phase of installation LCFG partitions the disk, creates file systems, sets up networking and installs the packages specified by the node profile over the network. As installation is an IO-intensive operation that makes extensive use of OS services, it provides a good test of VMM overhead.

We modified the LCFG install script to time the installation, and to log the duration to a file on the host OS. For our test, we installed a User Interface node under both Xen and User-Mode Linux. (There are approximately 700 software packages installed in this configuration.)

The results of these tests show that Xen was more than ten times faster than UML. The average installation time under Xen was 428 seconds (approx. 7 minutes), while a UML install took 4450 seconds (approx. 74 minutes). In comparison, an install onto a regular Linux kernel took 828 seconds (approx. 14 minutes). In this case the install server was accessed using the local network rather than internal networking, so the figures are not directly comparable. However as there is no alternative to the local network in a multi-computer gateway configuration, we feel this is a reasonable comparison to make.

4.2 Grid performance

We measured the performance of a number of typical command-line Grid operations on both the UML and Xen gateways. The `globusrun` test verifies that the user is authorised to run jobs on the CE, the `globusjobrun-nopbs` test submits a simple job which is executed directly on the CE, the `globusjobrun-pbs` test submits a simple job which is routed through the CE scheduler queue and executes on a WN, the `replica` test creates and deletes a replica of a 1 MB file on the SE and the `gridftpls` test lists the contents of the SE root directory. As the gateway servers are tightly coupled, these commands will exercise the whole system. For example, a job using replica management will be launched on the UI and queued on the CE before being executed on a WN to process data accessed via the SE.

We ran each of these commands fifty times on an otherwise idle system and recorded the mean duration. We ran the tests on both the VM-hosted UI and a UI running on Linux on a separate physical machine connected to the same switch: this allows us to observe the extent to which UI performance determines the overall responsiveness of the system.

Figure 2 summarises the test results — the y-axis displays the task duration in seconds. Four bars are shown for each of the tests: *UML-UML* and *Linux-UML* correspond to

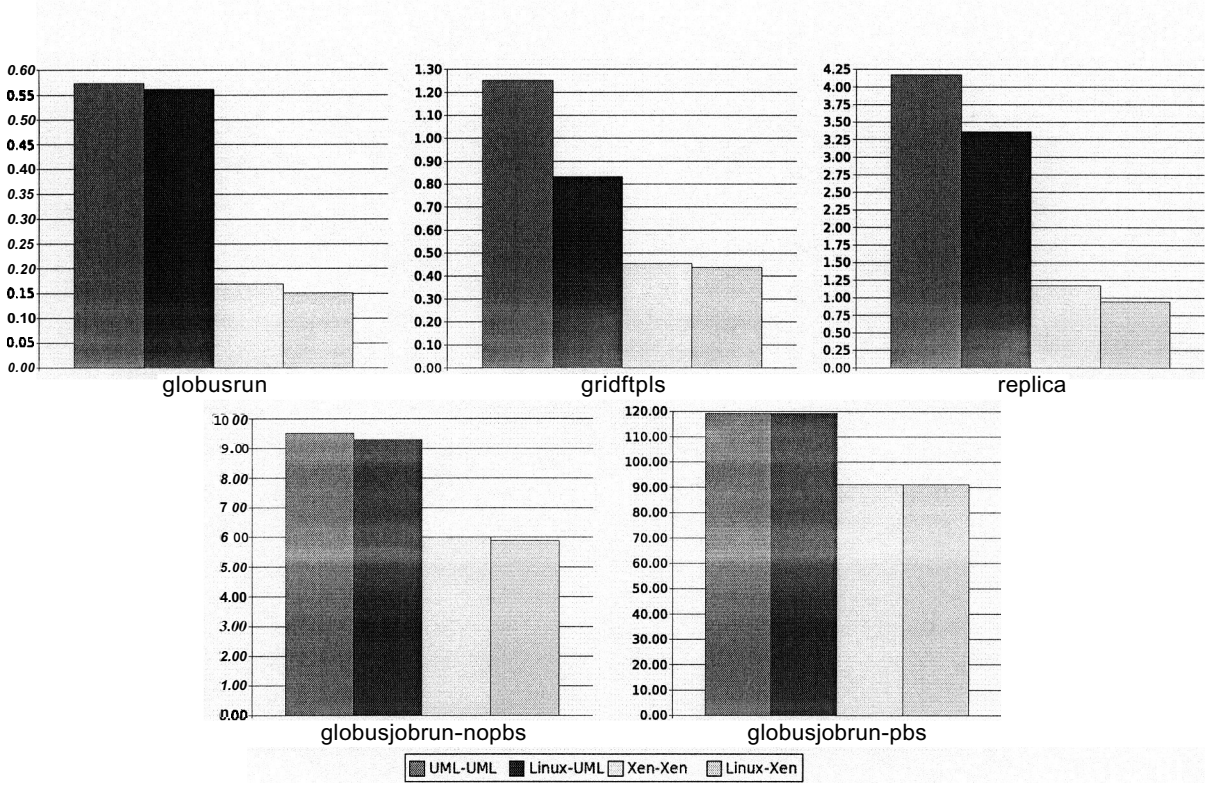

Figure 2. Performance of LCG operations

UML and Linux UIs accessing a UML gateway; ***Xen-Xen*** and ***Linux-Xen*** correspond to Xen and Linux UIs accessing a Xen gateway.

The results show that UML is consistently slower than Xen for the same operations. The `globusrun`, `gridftpls` and `replica` tests are between 2.8 and 4 times slower. The relative performance of the longer-running job submission tests is better: UML is approximately 60% slower when PBS scheduling is not used, and 30% slower when it is. However, even this represents 3 and 30 second delays for the user over the equivalent command execution on Xen.

We also have run informal tests on our local test Grid running Linux kernels. Results show that Xen performance is equivalent to that of the gateways hosted on Linux. Any performance overhead due to Xen is compensated for by the fact that intra-gateway communication uses internal networking rather than the local Ethernet.

4.3 Summary

The overheads introduced by UML are unacceptable for a production system. A standard site installation of four nodes per physical machine that would take a few hours on Xen would be a full day's work with UML! Interactive performance is also affected: users of the UI will experience sluggish performance even on an unloaded system, and this problem is exacerbated when multiple VMs are active on the same machine.

5 Using virtual machines

We have been using UML for the past year to run two gateway servers on the same machine; this configuration currently runs on five sites nationwide. As a result of the investigations described in this paper, we have decided to switch to Xen for the next phase of gateway rollout. The performance overhead due to UML, while just about acceptable for use on a single node, is too high for our target of five VMs per computer.

As we demonstrated in section **4.1,** UML is around ten times slower than Xen for OS-intensive tasks, making node installations and upgrades very painful. For interactive use, the UML UI feels sluggish compared to standard Linux — again, Xen does not suffer from this problem. Xen's architecture allows features that UML's user-space approach cannot provide: resource partitioning and hard isolation between VMs.

There are also management benefits: as the VMs' file systems are really just regular files on the host, we can easily back up an entire gateway by dumping the host file system. VMs also ease site installation as only one ma-

chine needs to be provided with network connection and power. Installation of individual servers is also more manageable. Even with network installation, unforeseen issues often arise that require physical access to the machines. With VMs this doesn't arise: once the host is up and running, all servers can be easily installed and accessed from the command line.

6 Related work

The work of Figueirdo *et al* [5] is complementary: they propose the use of virtual machines for Grid worker nodes whereas we use VMs for the gateway servers. Unlike us, they aim to support a variety of guest operating systems and so choose a VMM that supports full virtualisation. Other sites within the LCG collaboration have explored the use of VMs: the London e-Science Centre have used UML to provide an LCG-compatible environment on existing cluster machines [11], and Forschungszentrum Karlsruhe have used UML to host their install server [7]. To our knowledge, no-one else has implemented a complete site gateway using VMs. Outside the Grid community, the XenoServer [6] and Denali [12] projects both use VM techniques to support dynamically instantiated application environments for remote users.

7 Conclusion

We have demonstrated that it is feasible to construct a single-machine Grid gateway using virtual machines. However, our experiments show that the choice of VM technology is crucial. User-Mode Linux, while in widespread use, is impractical for our purposes due to its extremely high overhead for OS-intensive tasks. Xen, in contrast, performs well across a range of applications. Because Xen provides an OS environment that is indistinguishable from a regular OS instance, software servers can be run with the same configuration as on dedicated machines.

The use of VM technology has already brought management and deployment benefits in our site installations. The solution described here will allow rapid deployment of new gateways, enabling a significant increase in Grid participation. We believe that this approach will be of interest to many sites wishing to connect to the Grid for the first time.

Acknowledgements

We would like to thank the Xen team at the University of Cambridge for developing the Xen VMM and for providing useful support. Dell Ireland kindly donated the hardware. The original UML work drew on a configuration by David Coulson (available at `http://uml.openconsultancy.com`).

References

[1] Paul Anderson and Alastair Scobie. LCFG — the Next Generation. In ***UKUUG Winter Conference.*** UKUUG, 2002.

[2] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the Art of Virtualization. In ***Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles.*** ACM, 2003.

[3] S. Devine, E. Bugnion, and M. Rosenblum. Virtualization system including a virtual machine monitor for a computer with a segmented architecture. **US** Patent, Oct. 1998.

[4] Jeff Dike. A user-mode port of the Linux kernel. In ***Proceedings of the 4th Annual Linux Showcase & Conference, Atlanta.*** USENIX, 2000.

[5] Renato J. Figueiredo, Peter A. Dinda, and Jose A. B. Fortes. A Case for Grid Computing on Virtual Machines. In ***Proceedings of the International Conference on Distributed Computing Systems,*** May 2003.

[6] Keir A Fraser, Steven M Hand, Timothy L Harris, Ian M Leslie, and Ian A Pratt. The Xenoserver computing infrastructure. Technical Report UCAM-CL-TR-552, University of Cambridge Computer Laboratory, January 2003.

[7] Ariel Garcia and Marcus Hardt. User Mode Linux LCFGng server. `http://gridportal.fzk.de/websites/crossgrid/site-fzk/UML-LCFG.txt`, 2004.

[8] P. H. Gum. System/370 Extended Architecture: Facilities for Virtual Machines. ***IBM Journal of Research and Development,*** 27(6):530–544, Nov. 1983.

[9] Maxim Krasnyansky. Universal TUN/TAP Driver. `http://vtun.sourceforge.net/tun/`.

[10] LCG. LHC Computing Grid Project (LCG) home page. `http://lcg.web.cern.ch/LCG`, 2004.

[11] David McBride. Deploying LCG in User Mode Linux. `http://www.doc.ic.ac.uk/~dwm99/LCG/LCG-in-UML.html`.

[12] A. Whitaker, M. Shaw, and S. Gribble. Denali: Lightweight virtual machines for distributed and networked applications. In ***Proceedings of the USENIX Annual Technical Conference,*** 2002.