# 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. However, the hardware requirements for access to the Grid remain high: a standard Grid-Ireland 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-box Grid gateway that we have developed for nationwide deployment. We evaluate implementations of this architecture on two popular open-source VM solutions: Xen and User-Mode Linux (UML). Our results show that Xen outperforms UML for installation tasks and standard gateway operations. Performance is close to that of standard Linux, and more than acceptable for everyday use.*

*The Grid gateways support remote installation and management to facilitate integration with a national Grid infrastructure. 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 existing facilities to the Grid.*

## 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.

| | |
|---|---|
| **Install server** | configuration and software repository for all nodes |
| **Computing Element (CE)** | provides access to compute nodes |
| **Storage Element (SE)** | provides the data management interface |
| **User Interface (UI)** | access point for submitting jobs |

**Table 1. Essential Grid gateway servers**

Our national grid infrastructure is based on middleware being developed by the LHC Computing Grid (LCG) project [1]. LCG aims to integrate the capacity of worldwide scientific computing centres to enable processing and storage of large datasets produced by particle physicists. LCG provides a common software distribution and site configuration to ensure interoperability between widely distributed sites. The current release (2.2.0) uses LCFGng [2] to enable 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 sites. The Operations Centre provides information services (e.g. resource broker) to all sites nationwide. Each site hosts a Grid access gateway and a number of worker nodes that provide compute resources. While the access gateways are managed from the Operations Centre, the worker nodes are managed by local site administrators.

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 — this is the motivation for investigating the use of VMs.

A standard LCG gateway configuration makes significant hardware demands of a site. A minimum of four dedicated machines are normally required, each having a distinct function within the system (see Table 1).

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. This approach would maintain strong isolation between servers, and as each VM would appear to be a normal machine both to the server software and to external users, we would largely eliminate the need for any special configuration relative to the existing gateways.

## 1.2 Aims

Our aims may be elaborated as follows:

- *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.

- *Limit divergence from standard Grid site configuration:* We would like to maximise the amount of configuration data (LCFG profiles, software package lists) that would be common to multi-machine sites and single-machine sites.

- *Maintain central management of remote sites:* Gateway servers at remote sites are currently managed centrally using console redirection and Secure Shell (`ssh`) access. Any new setup must provide the same level of control.

- *Provide a simple installation process:* The installation process will be performed remotely and so must be robust and simple.

## 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 within a single OS instance. We now briefly list the advantages of our proposed approach over these alternatives.

### 1.3.1 Relative to multi-box solution

- *Lower hardware cost:* One server costs less than four servers!

- *Easier management:* A single machine solution makes fewer demands on site administrators in terms of machine room space, power requirements, network connections, etc.

### 1.3.2 Relative to single-OS solution

- *Isolation between logical servers:* Mainstream OSes provide weak isolation between user-level processes, making it relatively 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.

- *Similar configuration to multi-host sites:* 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 this hypothesis. In Section 2 we discuss the factors that determine our choice of VMM, in Sections 3 and 3.2 we describe the architecture of gateways built on two alternative VMMs, and in Section 4 we present performance measurements for both platforms. Section 5 relates observations derived from our use of VM technology in deployed sites, Section 6 discusses related work, and finally Section 7 summarises our findings and points to future work.

## 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

### 2.1.1 Technical

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

*Isolation:* In the standard configuration, each gateway server is hosted on a dedicated physical machine. With a single-box solution, the VMM should be able to provide isolation between the OS instances, so that even a catastrophic OS failure on one node will not bring down the others. Of course, a hardware failure will inevitably affect all hosted servers.

*Storage:* The logical servers each have different storage requirements. For example, the Storage Element should provide access to significant amount of storage for replication of potentially large data sets, while the Computing

Element does not require large amounts of storage. However, in our VM setup, these servers will all 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 an unacceptable 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 also hosts the User Interface, it must provide good interactive response times.

#### 2.1.2 Administrative

The VMM system 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

Virtualisation has been at the heart of operating systems development ever since the concept of virtual memory was introduced in the Atlas system [10]. A virtual machine is an abstraction that encompasses the entire computing environment, providing each user with an environment tailored to his applications and isolated from other users of the system. The virtual machines are controlled by a supervisor program, or 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 [9]), where it was used to allow many users to share the resources of a single large computer.

Recently, the sustained increase in processor speeds has revived interest in implementing VMMs on commodity hardware; the past few years have seen a steady trickle of new commercial and open-source VMMs which provide varying levels of virtualisation.

Full virtualisation virtualises a complete instruction-set architecture: any OS that will run on the underlying hardware will run on the VM. Para-virtualisation presents a modified interface to guest OSes: they must be ported to the new VM "architecture". System call virtualisation provides an application binary interface that enables guest OSes to run as user-space processes.

VMWare [4] is a commercial product that provides full x86 virtualisation on both Windows and Linux. Xen [3] uses para-virtualisation and currently supports Linux, FreeBSD and Windows XP ports. User Mode Linux (UML) [5] 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.

### 2.3 Choices

In our evaluation, we will focus on Xen and UML. These are both open-source projects, meaning that it will be possible for us to customise the code if we need to. They are also stable projects that have already been deployed and they have active user communities to provide support. We have excluded commercial VMMs from our experiments due to cost considerations and licensing issues.

Because Xen provides only para-virtualisation, operating systems must be ported to run on the Xen "architecture". However, this is not a problem here as the LCG server software is targeted for Linux platforms. It is easy to compile custom kernels with Xen support, and the LCFG profiles can then be modified to selectively install these kernels on the correct machines.

UML is already a popular solution for VM applications including virtual web servers, shell accounts, security 'honeypots' and many others. UML VMs can be run on an unmodified Linux kernel, although specially patched kernels may be used to improve performance. Again, there are no fundamental difficulties with running the LCG software on UML.

## 3 Gateway architecture

The basic architecture is the same under both Xen and UML: the host OS runs an LCFGng [2] 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

#### 3.1.1 VM setup

Each VM is allocated 200 MB of memory and 20 GB of disk space.

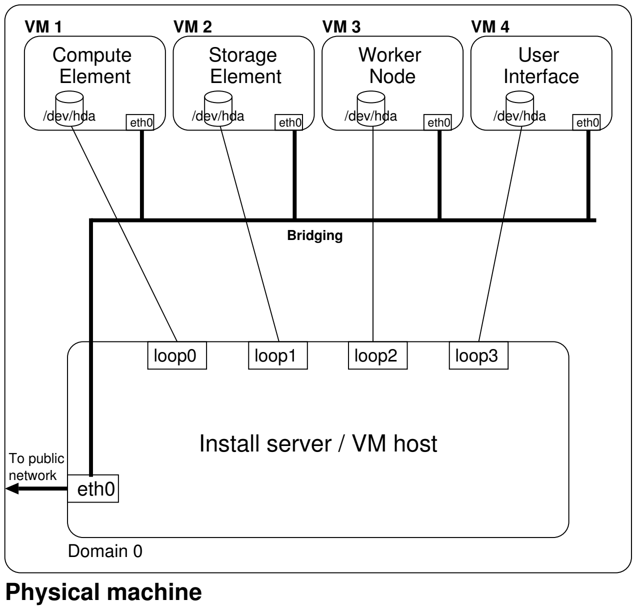

**Figure 1. Architecture of grid gateway**

### 3.1.2 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.

### 3.1.3 Storage

The file systems for the VMs are backed by sparse files on the host file system. Sparse files are used since they allow the hosted file systems to grow. 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.

### 3.1.4 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.). The profile for each server is closely based on the standard node profile. Changes necessary for Xen are included from a separate file thus reducing the impact of the necessary modifications.

### 3.1.5 Control of VMs

We created a control script on the server for starting and stopping the VMs. It has two modes (install and boot); both the mode and the choice of node are specified using command-line parameters. When it is invoked, the script locates the configuration file for the node, and then uses the information stored there to generate the appropriate command line for the Xen domain creation tools.

When running in install mode, the script first creates a sparse file and attaches the appropriate loopback device to it (using the file size and device name stored in the configuration file). The Xen domain is then booted using 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 loopback device is attached to the backing file, and the Xen domain is booted from the appropriate partition on the device.

### 3.1.6 Remote management

Each gateway must support remote management so that Grid-Ireland staff can initiate and control upgrade procedures. The server hardware includes a dedicated Ethernet management interface that provides console redirection from system boot time. This allows BIOS parameters to be modified if necessary.

Consoles for each of the server VMs can be accessed from the host OS. 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.

### 3.2.1 VM setup

Again, the host OS runs an LCFG install server with each of the nodes running in its own VM. In this case, the VMs are actually standard user-level processes. 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.

### 3.2.2 Networking

Networking is provided using the Linux TUN/TAP [11] software, 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.

### 3.2.3 Storage

Each partition (root, boot and swap) is backed by a file on the host computer's file system under the hierarchy `/var/umlinux/`*`node`*`/fs`.

### 3.2.4 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.

There are two levels of configuration data: parameters common to all nodes (e.g. LCFG server name) are stored in `/etc/sysconfig/umlinux`, while node-specific parameters data (e.g. VM memory allocation) are stored in the `/var/umlinux/`*`node`*`/config` files.

### 3.2.5 Control of VMs

As with Xen, we have developed control scripts to install, start, and stop the VMs. These scripts automate the creation of file systems, configuration of networking and other tasks needed before VM execution can start. A control script is used to start and stop the VMs. This script is first called at system startup to initialise the UML networking. The same script may be used to to start and stop VMs. On a VM startup call, the script reads configuration data stored for that VM (identified by its node name), sets up networking for that node and then starts a UML process with the appropriate command line parameters.

### 3.2.6 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 applications specific to our target environment; a detailed comparison of UML and Xen performance on standard benchmarks may be found in [3].

We ran all the tests on the same machine: a Dell PowerEdge 1750 server with two 2.4 Ghz processors, two 140 GB SCSI disks and two Gigabit Ethernet cards. (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.

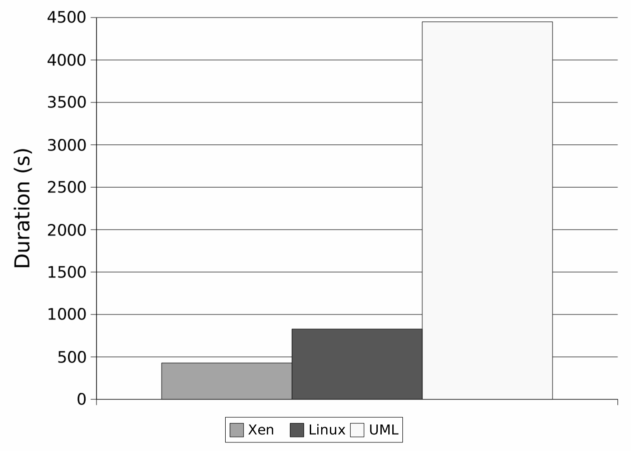

**Figure 2. LCFG installation of UI node**

## 4.1 Performance

### 4.1.1 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, installs the packages specified by the node profile, and sets up networking. 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 record timestamps at the start and end of installation, and to log the duration of the installation 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.)

Figure 2 shows the results of these tests. Xen was more than ten times faster than UML: the average installation time under Xen was about 7 minutes, while a UML install took approximately 74 minutes. In comparison, an install onto a regular Linux kernel took around 14 minutes. Of course in this case the install server was being accessed over the local network rather than through internal networking, so the figures are not comparable. However as there is no alternative to the local network in a multi-computer gateway configuration, we feel this is a fair comparison.

|  |  |
|---|---|
| `globusrun` | Verify that the user is authorised to run jobs on the CE |
| `globusjobrun-nopbs` | Submit a simple job which is executed directly on the CE |
| `globusjobrun-pbs` | Submit a simple job which is routed through the CE PBS queue and executes on a WN |
| `replica` | Create and delete a replica of a 1 MB file on the SE |
| `gridftpls` | List SE root directory contents |

**Table 2. LCG test operations**

### 4.2 Grid performance

The gateway servers provide the primary interface to Grid services for a site and their performance determines the quality of the users' experience. At the current stage of Grid deployment, administrators face an ongoing challenge to persuade users of existing clusters to migrate to the Grid. This task will be made more difficult if the gateway services are unresponsive.

We measured the performance of a number of typical command-line Grid operations on both the UML and Xen gateways. Operations included authentication checks, job submission, file replica creation and directory listing commands (full list in Table 2 — for more details on the individual commands see [13]). As the gateway servers are tightly coupled, these commands will tend to exercise the whole system. For example, a command executed on the UI will invoke an operation on the CE, which may need to query the SE before it returns a result.

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 a separate physical machine connected to the same switch: this allows us to observe the extent to which UI performance determines the overall responsivenes of the system.

Figure 3 summarises the test results. Four bars are shown for each of the tests: *UML-UML* and *Linux-UML* correspond to 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 that runs standard Linux kernels. Results show that Xen performance is roughly equivalent to that of the gateways hosted on Linux. This is what we would expect: any performance overhead due to Xen would be compensated for by the fact that intra-gateway communication uses Xen's internal networking rather than the local Ethernet.

### 4.3 Resource usage

The level of extra resource usage due to the use of a VMM can determine whether it is practical to run the required number of servers on a single machine. For example, if the VMM management software demanded a large proportion of system memory, it would severely limit the performance of the servers it was hosting by reducing the amount of memory available to them.

Xen doesn't impose any significant memory overhead: because it doesn't have to maintain shadow page tables for each VM it keeps only 20 kB of state per domain ([3], p. 12). UML also has a low overhead as the VMs are in actual fact normal user-level processes.

### 4.4 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 logging into the UI 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

### 5.1 Experience

We have been using VMs for the past year. In our current production setup, we use UML to double up usage of some of the gateway servers (for example, the User Interface server is hosted on the Storage Element machine). This configuration runs on five sites nationwide.

However, as a result of the investigations described in this paper, we have now 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, has proved too high for our target of five VMs per computer.

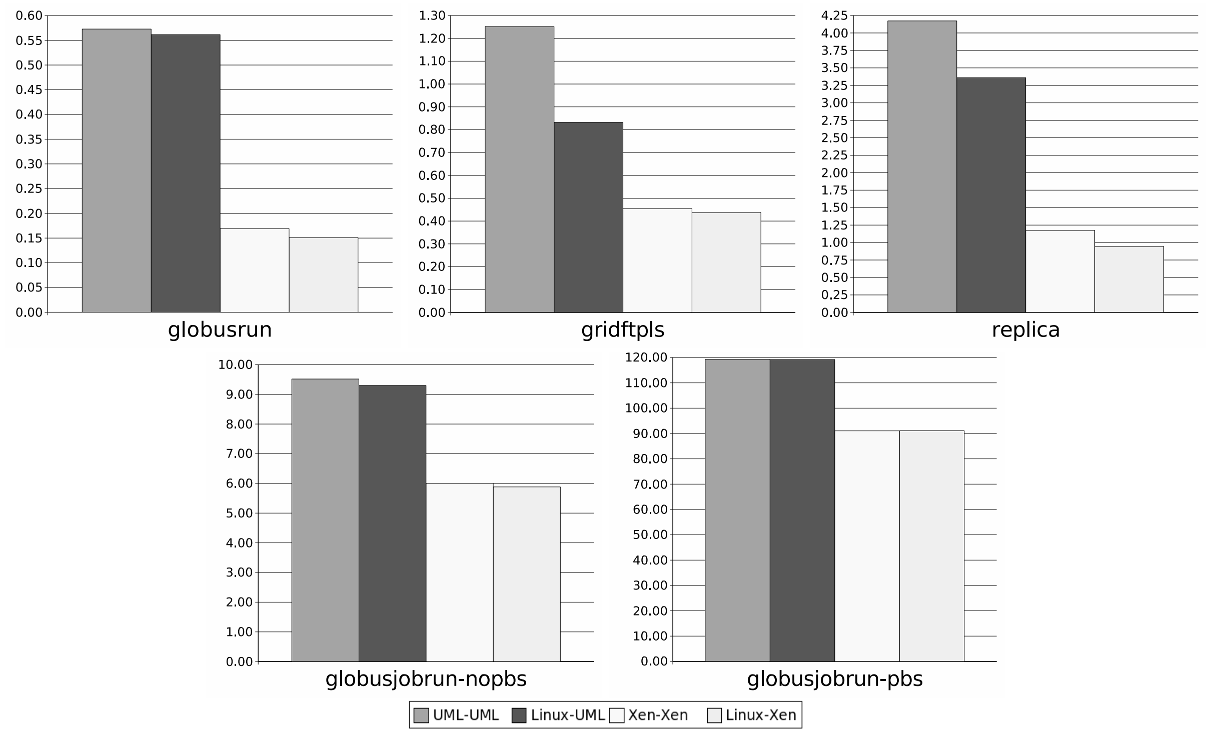

**Figure 3. Performance of LCG operations**

As we demonstrated in section 4.1.1, UML is around ten times slower than Xen for OS-intensive tasks, making node installations and upgrades very painful. The interactive response of the UML UI feels sluggish compared to standard Linux — again, Xen does not suffer from this problem.

In addition, Xen's architecture enables features that UML's user-space approach cannot provide: for example, resource partitioning and hard isolation between VMs. We have found it easy to extend the Xen management tools by calling them from control scripts.

### 5.2 Management benefits

As the VMs' file systems are hosted on regular files on the host machine, an entire site may be easily backed up by dumping the file system of the host.

Installation of a site is much easier with VMs. Firstly, only one machine needs to be provided with network connections, power, etc. Secondly, installation of individual server VMs is simpler. We have discovered that even when LCFG and PXE booting are used to remotely install machines, there are often BIOS issues that require someone to go round each of the machines. With VM installation, these issues don't arise: once the host machine is up and running, all server nodes can be easily installed from the command line.

## 6 Related work

The work of Figueirdo *et al* [6] is complementary: they propose the use of virtual machines for Grid worker nodes whereas we use VMs for the gateway servers. They aim to support a variety of guest operating systems and so choose a VMM that supports full virtualisation. Outside the Grid community, the XenoServer [7] and Denali [14] projects both use VM techniques to support dynamically instantiated application environments for remote users.

The LCG-2 team are modifying the software distribution to reduce the number of separate server machines needed. However, we believe that VM solutions will continue to be useful as they provide an effective means of partitioning complex sets of services into distinct logical units.

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 [12], and Forschungszentrum Karlsruhe have used UML to host their install server [8]. To our knowledge, no-one else has implemented a complete site gateway using VMs, and with such a low performance overhead.

## 7 Conclusion

We have demonstrated that it is feasible to construct a single-machine Grid gateway using virtual machines. However, the choice of VM technology is crucial. User-Mode Linux, while popular, is an impractical solution due to an extremely high overhead for OS-intensive tasks. Xen, in contrast, performs well enough to be indistinguishable from a real machine.

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. In addition, strong isolation of VMs allows logically distinct servers to run safely on the same machine.

We have already experienced the benefits of VM technology in our site installations. It has allowed us to make more efficient use of machines by doubling up servers. Running full gateways on VMs will enable us to achieve a significant increase in Grid participation on a national level, and will be of interest to many sites wishing to participate in Grid-based research 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 making it freely available. They also provided useful support during the development of this system.

We would also like to thank Dell Ireland who provided the hardware for the development and national deployment of this system.

The original work on UML drew on a configuration published by David Coulson (available at `http://uml.openconsultancy.com`).

## References

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

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

[3] 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 Press, 2003.

[4] 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.

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

[6] 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 (ICDCS)*, May 2003.

[7] 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.

[8] Ariel Garcia and Marcus Hardt. User Mode Linux LCFGng server. `http://gridportal.fzk.de/distribution/crossgrid/crossgrid/wp4/sites/fzk%/uml-lcfg/UML-LCFG.txt`, 2004.

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

[10] T. Kilburn, D. J. Howarth, R. B. Payne, and F. H. Sumner. The Manchester University Atlas operating system part I: internal organization. *The Computer Journal*, 4(3):222–225, 1961.

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

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

[13] Antonio Delgado Peris, Patricia Mndez Lorenzo, Flavia Donno, Andrea Sciab, Simone Campana, and Roberto Santinelli. LCG-2 user guide. LHC Computing Grid Manuals Series. Available via `https://edms.cern.ch/file/454439//LCG-2-UserGuide.pdf`, September 2004.

[14] 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.