GridBuilder: A tool for creating virtual Grid testbeds

Stephen Childs Brian Coghlan Jason McCandless
Department of Computer Science
Trinity College Dublin
Ireland
Email:{childss,coghlan,mccandjm}@cs.tcd.ie

Abstract

Grid software developers and Grid site administrators both require realistic testbeds where they can test applications and middleware before deployment on production infrastructure. Such testbeds should be dynamically reconfigurable to allow replication of real-world configurations. While the combination of service nodes needed for a particular application may vary greatly, there is a fixed set of node types which are used frequently: these are the building blocks needed to construct testbeds.

We present GridBuilder: a web-based virtual machine (VM) manager that supports the rapid creation and customisation of Grid nodes based on standard configurations. GridBuilder allows users to create a library of filesystem images and then generate independent filesystems based on these images. Copy-on-write is used for fast initialisation and efficient use of disk space. GridBuilder uses standard Grid configuration tools to automatically configure the middleware on new Grid service nodes. Users can save and restore snapshots of nodes and can import images to create new node types.

1 Introduction

The use of virtual machines is becoming widespread in the Grid community due to the availability of free virtual machine monitor (VMM) software such as Xen [1] and VMWare Player [7]. There are many compelling applications for VMs on the Grid: deployment of application-specific services [8], dynamic creation of execution environments on compute nodes [10, 12], deployment of Grid service nodes [5], and the creation of replica Grids for training and certification testing [3, 6].

In this paper we focus on the creation of replica Grid testbeds. Our own experience with VM-based testbeds has revealed the need for management tools to automate the process of creating and configuring virtual machines. In particular we have observed that while each user may require her own custom testbed, customisation is the result of different selections from a relatively small set of standard node types. The particular combination of node types will depend on the nature of the testing being performed. For example, when stress testing a batch system, it may be necessary to create a large virtual cluster of compute nodes, whereas for development of a match-making service, a testbed containing a selection of differently-configured resources and sites is required. If we can automate the creation of these standard node types while allowing the customisation of site-specific parameters, a huge variety of testbed configurations can be created easily.

The design of GridBuilder has been driven by the demands of our own research environment, and so we now describe the context of our work.

1.1 Context

Our research group at Trinity College Dublin runs the Operations Centre for Grid-Ireland, the organisation that provides Grid services to research institutions in Ireland. The Operations Centre is responsible for managing Grid gateways at member institutions, and for validating and deploying releases of Grid middleware, a task that requires thorough testing procedures. Our group also develops new Grid middleware: developers need to test potentially disruptive software in a realistic environment that is isolated from production systems.

We have developed a test environment that replicates our national Grid infrastructure within an isolated network [6]; this provides a good platform for administrators to test configuration changes and for developers to test new software against a realistic infrastructure. The testbed includes both dedicated replica machines hosting a single OS as well as VM server machines that host multiple OSes. Although we have dedicated a significant number of machines for testing, there is a continual demand for more. By using virtual machine technology, we can run multiple OSes on each of the

physical machines within the testbed, greatly increasing the number of nodes available.

Our test environment allows considerable flexibility: the widespread use of virtual machines makes it inexpensive to set up a new node, the isolated network allows the use of "real" IP addresses, and the existence of replica central Grid servers means that users can plug their test nodes into a pre-existing infrastructure. However, this flexibility results in a high burden on the user: in order to set up a new node, a user must know how to configure filesystems, VMs, Grid middleware, and Linux networking. This requires familiarity with a large number of different tools.

Our testbed has a wide spectrum of users, from experienced site administrators familiar with the intricacies of configuring Grid middleware, to Java middleware developers who just need a working site configuration to test their new software on. We want to automate the routine tasks associated with creating new Grid nodes in order to reduce the amount of time that users spend configuring the same basic set of nodes again and again. If the basic components (Grid nodes) can be generated quickly and easily, developers will have more time for developing and testing their software.

1.1.1 EGEE middleware

Our work takes place in the context of the world-wide production Grid created within the Enabling Grids for E-SciencE (EGEE) project. In the EGEE model (as described in [4]), a Grid site is composed of instances of basic node types: the Computing Element (CE), which provides an interface to compute resources via Local Resource Management Systems (LRMS); the Storage Element, which provides an interface to local storage; the User Interface (UI), which hosts client software for accessing the Grid; and Worker Nodes (WNs), on which users' jobs are run. A site may have one or more of each of these node types. The middleware on a node is usually configured using YAIM [9], a suite of shell scripts that set up the middleware according to a site-wide configuration file. The functionality of Grid sites is tested by the regular execution of test jobs (Site Functional Tests) that execute commands verifying a wide range of Grid operations.

Although we focus on the EGEE model in our current implementation, support for other Grids could easily be added, as the approach of defining standard node types should be widely applicable. For example, we also maintain a testbed running Globus Toolkit 4 and Condor; we could integrate this into GridBuilder by creating node types "GT Gatekeeper", "Condor Manager" and "Compute Node".

1.2 Aims

1.2.1 Ease of use

The principal aim is to make it easy for any user to create standard test nodes (or groups of test nodes) according to her own requirements. We must abstract the intricacies of Grid site configuration, and by default, the system should only prompt users to make high-level choices that they can understand (e.g. "Create worker node configured for the Trinity College Dublin site"). This is most easily achieved by storing detailed information about the configuration of standard node types and sites, using formats from standard fabric management tools. A web-based interface will provide a familiar front-end to the system.

1.2.2 Performance

The system must be responsive under typical testing loads if it is to be useful. The first priority is the selection of appropriate VM technology that can deliver virtualisation without unacceptable overheads. It is also important to avoid the overhead of copying whole filesystem images; copy-on-write techniques should be used when cloning system images.

1.2.3 Security

Although this system is designed to run in a secure test environment, we still need to protect users from each other. Users who create VMs need to be sure that others cannot interfere with their configuration. Users must be authenticated and authorised and ownership of VMs enforced. Of course, administrators will have appropriate powers to intervene.

1.2.4 Configuration using standard Grid tools

This system aims to automate the process of creating Grid testbeds composed of standard Grid nodes. This implies the need for integration with standard tools used to install and configure Grid nodes. When a user requests the creation of a particular node type, GridBuilder should invoke standard tools to install and configure appropriate software.

1.2.5 Versioning of VM images

Using virtual machines makes it easy to take snapshots of entire OS instances. This allows users to try a particular line of enquiry, installing and modifying software on a node, with the confidence that they can return to a previous snapshot if that approach proves fruitless. Our system supports such use cases by recording timestamps with created filesystems; each snapshot also has a user-defined description to allow identification at a later date.

1.3 Overview

In Section 2 we present a high-level view of the system architecture, in Section 3 we describe the details of our implementation. We then describe usage scenarios in Section 4. In Section 5 we discuss related work and finally we present our conclusions in Section 6.

2 Architecture

In this section, we describe the architecture of GridBuilder at a high level so that the basic building blocks can be seen clearly without implementation details intruding.

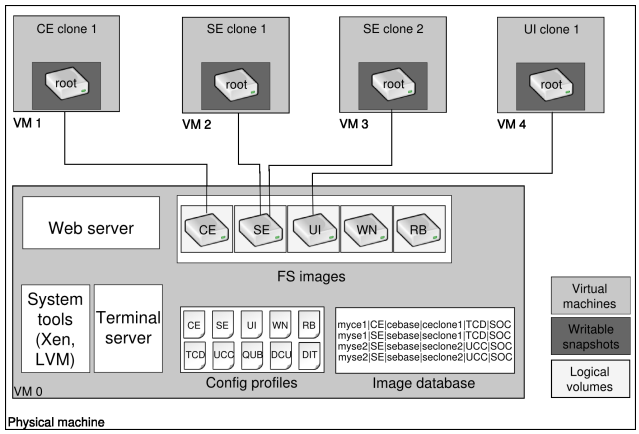

Figure 1. Architecture of a VM server

Figure 1 shows the architecture of a GridBuilder server. On each server, a manager VM (VM 0) hosts the tools needed to control and configure the other VMs. The web server that provides the user interface also runs on the manager. While not strictly accurate, it may be helpful to think of the manager VM as being a host for the other VMs.

In this example, the VM server is hosting two Storage Element (SE) VMs, one Compute Element (CE) VM and one User Interface (UI) VM. The filesystems of both SE VMs point to the same base image on the VM host filesystem; however, as these are copy-on-write (COW) snapshots, the VMs can read and write independently.

We have been using VMs without having an automated system for creating VMs and managing filesystems. Users were spending a lot of time manually creating, duplicating and managing file systems for new VMs. They also found the configuration of networking and Grid middleware for the VMs to be complex, and the need to master many different command-line tools in order to create and manage simple VMs was frustrating. This experience led us to create GridBuilder.

To address the filesystem management problems, each server stores a library of filesystem images for standard node types. This allows users to set up new VMs without having to create new filesystems. Rather than duplicating these base images whenever a new VM is created, we create copy-on-write overlays. This allows new VMs to be started rapidly without the overhead of copying an entire filesystem, and also makes efficient use of storage as each new node only requires space for changes it makes relative to its base image.

A typical library of images would include standard installations of Grid service nodes (CE, SE, UI, etc.) and cluster nodes (WN) as well as default installations of various Linux distributions. This initial library can be expanded; we provide facilities for users to import new base images from external filesystems they have created. Once a VM has been created, its filesystem persists even after the VM has been shut down. Users can therefore return to old versions of a particular VM, or create multiple versions with different software releases or configurations.

To address the difficulty of configuring new VMs, we automate the setup of networking and Grid middleware. Configuration profiles are stored for a set of various replica sites and are applied before a new VM is started. Integration with existing configuration tools is also crucial: for example, we currently support the Quattor [13] fabric management tools used at a number of Grid sites, and YAIM, the EGEE standard configuration tool. GridBuilder invokes these tools to configure Grid services based on standard profiles without any need for the user to intervene.

The user interface allows high-level configuration in terms users can understand and hides the complexity of the various underlying tools. It is implemented as a web-based interface rather than a stand-alone application to allow easy accessibility from other machines. System management tools on the VM server are called directly from the back-end code.

3 Implementation

3.1 Virtual machine technology

Our VM servers run the Xen virtual machine monitor (VMM) [1]. In contrast to other common VMMs which fully virtualise the x86 processor, such as VMWare [7] and QEMU [2], Xen is a para-virtualised system. A low-level micro-kernel (known as the hypervisor) runs on the bare hardware and provides an interface through which hosted virtual machines can access low-level services. Guest operating systems must be modified to use the hypervisor: this is comparable to porting a kernel to a new architecture. The principal advantage of para-virtualisation is lower performance overheads; the main drawback is that operating sys-

tems must be ported to the VM platform. However, this is not a problem in many server environments where a limited range of guest operating systems is required. In our case, Linux is the primary guest OS for Xen, and as Linux is a de facto standard within the EGEE Grid community, Xen meets our needs well.

3.2 Storage of images

We use the Linux Logical Volume Management (LVM) [15] software to provide a pool of storage for filesystem images. We allocate a large partition to the LVM volume group; if further space is needed new disks or partitions can simply be added to the volume group. Each base filesystem image is stored on a separate logical volume. The base images are created by copying a filesystem from a previously-installed node onto a newly created logical volume of an appropriate size.

The LVM software supports snapshots of logical volumes for backing up system state. A recent development is the addition of writeable snapshots which effectively provide copy-on-write (COW) functionality. When a user requests creation of a new VM, GridBuilder creates a writeable snapshot of the relevant base image and uses it as the root filesystem for the new VM.

3.3 Grid configuration

When a new Grid node is created, the middleware must be configured with details appropriate to that node. For example, networking parameters will vary according to the hostname of the node, and each site will support different virtual organisations. Our approach is to integrate support for existing Grid configuration tools: GridBuilder loads or generates site configurations and then invokes these tools directly.

We currently support the two main methods of Grid middleware configuration used within EGEE: Quattor [13], a comprehensive fabric management system which configures OS components as well as Grid middleware, and YAIM [9], a suite of shell scripts.

The benefit of complete fabric management systems such as Quattor is that a single configuration profile encapsulates the vast majority of system state for a Grid node of a particular type. By applying a profile to a base OS installation, it is possible to regenerate a node completely.

In theory, it should be possible to rely on fabric management clients to fully configure a node based on its hostname and associated configuration profile. In practice, we have found it useful to perform a number of pre-configuration steps on the VM's filesystem before booting it. Firstly, we mount the filesystem on the VM host, and reconfigure the static network settings in the usual locations. (N.B. if DHCP is being used, then this step is not necessary.) We also transfer the new profile to the VM filesystem and using `chroot` and compile the profile in place. The result of this pre-configuration is that when the VM boots for the first time, the client already has an up-to-date profile and can immediately apply any changes necessary (e.g. installing or removing packages).

We have gathered a set of Quattor profiles and network configurations for Grid-Ireland sites, and these form part of the GridBuilder distribution. When a user requests a new node within a particular site (for example, TCD), the relevant network configuration is copied into place and the configuration profile for a TCD node is initialised. For YAIM, existing site configuration files can be uploaded, or a configuration can be generated based on parameters entered by the user (e.g. hostnames of service nodes). As YAIM includes both installation and configuration, a new Grid node can be created easily from a plain OS installation.

3.4 VM database

GridBuilder uses a database to keep track of imported base images and to store the VM configurations to allow future invocations. When a new base image or VM configuration is created, the details are stored in a database table. Table 1 shows some sample entries in the VM table. Status pages showing available VMs and associated filesystem images are generated from the database. The database is stored in a MySQL server on the GridBuilder server. Entries are inserted directly into the database using MySQL bindings for Python.

3.5 User interface

The prototype user interface is implemented using web pages generated by CGI scripts written in Python that invoke Xen and LVM commands directly. Figure 2 shows the screen for creating a new VM.

The user can choose which site their new VM will be associated with. This choice determines which base configuration will be used for network addresses and profile compilation. The user can also specify the amount of memory allocated to the VM, the hostname and IP address for the VM and the amount of space allocated for changes in the COW file system. GridBuilder then displays the results of the various operations performed: filesystem creation, network preconfiguration, profile compilation and VM startup. The user can then access the console of the new VM as described in section 3.8.

A list of base node images stored in the database is displayed. In this example a clean installation of Scientific Linux 3.0.7 is available `sl307`, as well as Grid nodes of type CE (`sl307-glite-ui-3.0.2`) and UI

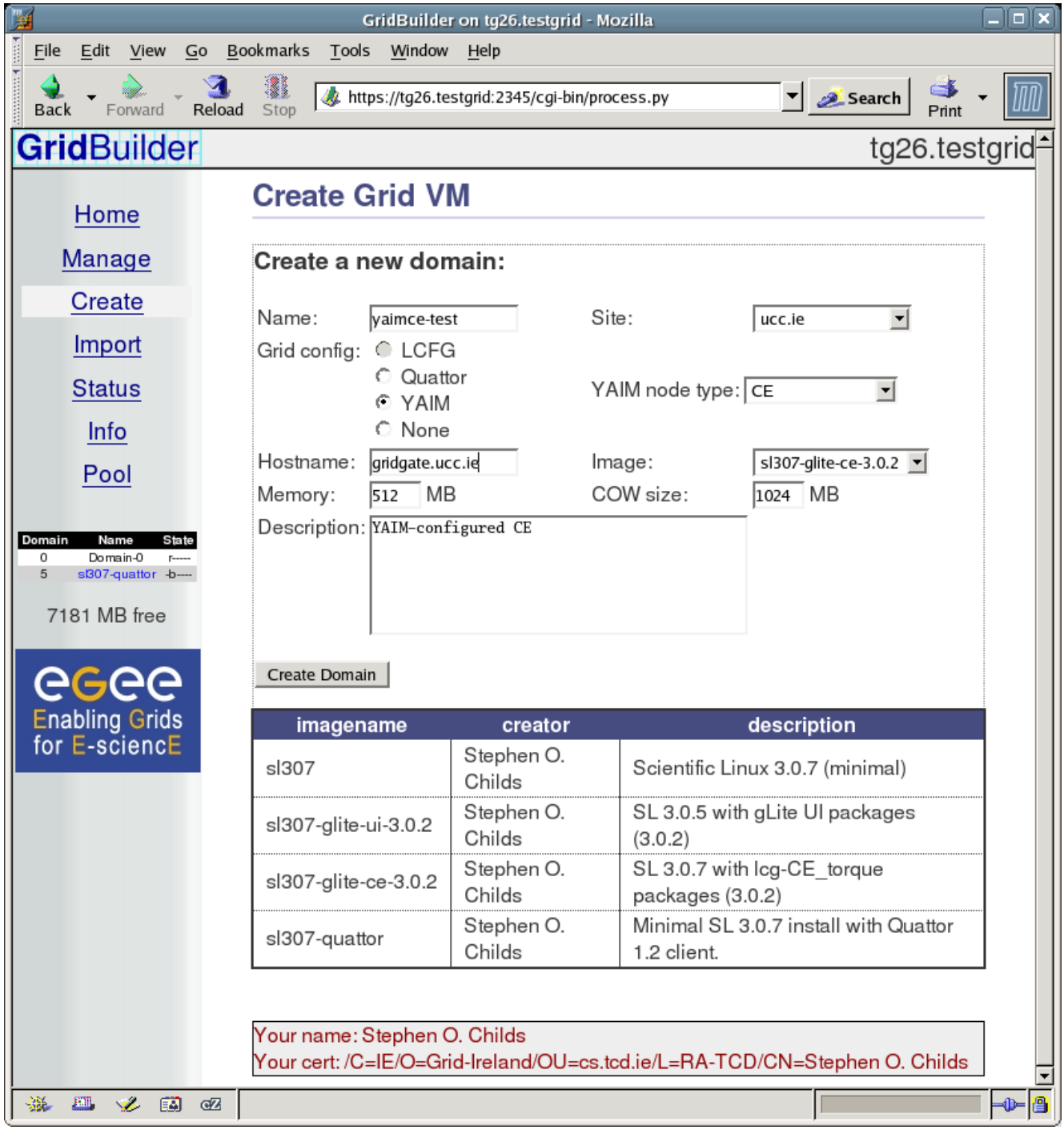

Figure 2. Interface for VM creation

Node name	Base image	COW image	Saved	Site	Owner
gridstore_grid_ul_ie	gridstore	gridstore_gridstore_grid_ul_ie	no	grid.ul.ie	Stephen O. Childs
gridmon_it-tallaght_ie	gridmon	gridmon_gridmon_it-tallaght_ie	no	it-tallaght.ie	Stephen O. Childs
gridmon_ait_ie	gridmon	gridmon_gridmon_ait_ie	no	ait.ie	Stephen O. Childs
gridui_dit_ie	gridui	gridui_gridui_dit_ie	no	dit.ie	Jason McCandless

Table 1. Sample entries in VM database

`sl307-glite-ce-3.0.2`, and a node with Quattor client software installed (`sl307-quattor`). Users select the type of node they wish to create and the system creates a COW file system from the appropriate base image.

A similar screen exists for managing existing VMs. It displays both active and inactive VMs and provides controls for starting, stopping, checkpointing, pausing and resuming them. GridBuilder associates timestamps and descriptions with each filesystem image, allowing users to store and manage multiple snapshots of nodes in different states.

The user interface also allows users to import filesystems to serve as base images. Various formats are supported as users tend to have different techniques for creating filesystem images. Existing snapshots can also be used to create new base images. The import operation takes some time as in this case the data must to be copied into a new logical volume. The `rsync` tool is used to minimise the amount of data copied.

3.6 Security

Access to the web interface is secured using GridSite [16], a module for the Apache web server that provides access control based on authentication of users by Grid certificates. Text-based access control lists can be set up to restrict access to the GridBuilder interface to users with appropriate certificates. GridSite also makes the distinguished name of the authorised user available to the web front-end, allowing GridBuilder to identify the current user and to record the ownership of VMs in the database. We currently use this information to prevent users from deleting images they don't own, however we plan to implement a more comprehensive authorisation model in the future.

3.7 GridBuilder pool management

A Grid testbed will normally contain multiple GridBuilder hosts. It is helpful for users to see the state of all hosts within a testbed, as then they can make informed decisions about where to host new VMs. GridBuilder includes a tool that queries the Xen managers on all configured hosts, and generates a page with direct links to the GridBuilder management pages for remotely-hosted VMs. The list of GridBuilder hosts is currently statically defined; in the future we hope to provide a registration system whereby new GridBuilder hosts can notify a central server of their existence automatically.

3.8 Console access

It is important that users can access and control the VMs they create at console level. For example, this would allow the boot process to be monitored so problems can be detected. Console access also enables access to a VM whose networking has been misconfigured. For maximum convenience, console access should be provided through the same web interface used to configure VMs, to avoid dependence on external programs. We modified AjaxTerm [14], a web-based terminal application, so that it can read directly from the console of a Xen VM. Each VM has an associated AjaxTerm, which is created during the setup phase of a new VM. The AjaxTerms run on ports allocated according to the ID of the VM (for example the terminal for VM 5 is accessible at port 9005). An AjaxTerm client is built into the management page of the VM and provides full console access.

4 Using GridBuilder

4.1 VM lifecycle

Figure 3 shows a sample procedure used by a developer to populate GridBuilder with the node types she requires. Initially GridBuilder contains a single base image, Base A, which is a default Scientific Linux 3 installation. The developer then requests the creation of two VMs with filesystems based on Base A: the system creates these as COW snapshots (which are initialised very quickly).

The developer uses the first VM (COW A1) to install and configure worker node software; this goes according to plan and results in a new candidate base image. The developer then requests that this image be imported as a new SL3 WN base image (Base B), identified as SL3WN.

The developer uses the second VM (COW A2) to install and configure computing element software; this attempt isn't successful, and so the VM is shut down without making a new base image from it. However, if she wishes

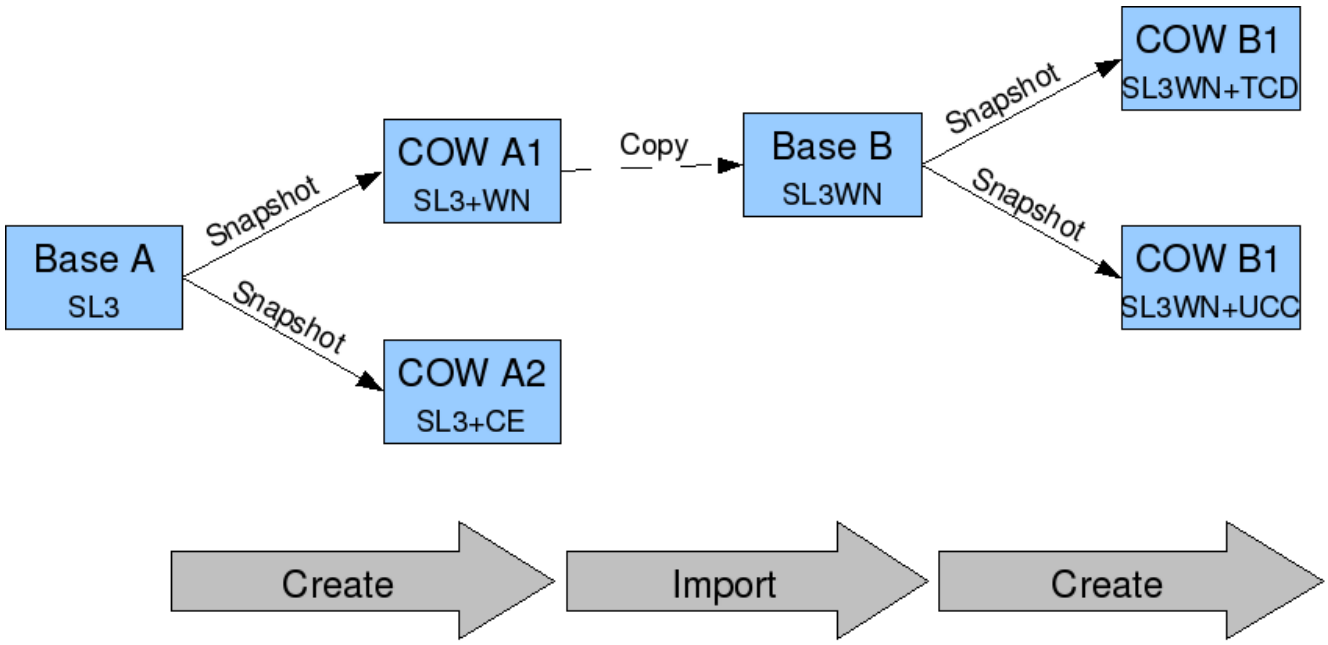

Figure 3. Typical lifecycle of VM images

to revisit that VM in the future, the fi lesystem still exists on the disk and can be restarted.

The image library now contains two base images: Base A and Base B. The developer now uses the SL3WN image as a base for two new VMs: COW B1 which is a WN confi gured for the TCD site, and COW B2 which is a worker node confi gured for the University College Cork (UCC) site. Potentially, each of these could be saved as new base images, allowing for the rapid creation of a whole cluster of TCD WNs for example.

4.2 Performance

The combination of Xen and LVM snapshots provides excellent performance: it takes approximately 6 seconds to create a new FS image and confi gure a new VM on a dual-processor 2.8 GHz PC (Dell PowerEdge 1450). Of course, the user must then wait for the normal boot process to complete before logging in; for a minimal Scientifi c Linux 3.0.7 installation, a login prompt was available in approximately 25 seconds. Alternatively, a saved VM can be resumed to a fully running state in a few seconds.

4.3 Use cases

A primary use case is deployment testing: Grid system administrators need to test new middleware releases in an environment that closely resembles the real infrastructure. As GridBuilder stores images of standard infrastructure servers, new virtual sites can quickly be initiated. These virtual sites can then be upgraded and tested to reveal any problems that would otherwise disrupt service. Because the site networking information is a parameter of node creation, a variety of sites can be created quickly. The ability to track particular snapshots is also useful when comparing behaviour with different software versions.

Another use case is specialised testbeds for middleware development. Using GridBuilder, developers can quickly instantiate new client and server nodes to work on. When a series of incremental changes is necessary, developers can save the state of the node and resume it with all of their changes intact. When they need to start again with a clean system (e.g. to test installation procedures), developers can quickly create fresh VMs based on the base fi lesystem image for the node type. This is of course all possible without our tool, but automatic management of images relieves the developer of a great deal of work.

Another good application of the tool is the creation of tutorial testbeds. Pre-confi gured images stored on the VM server can be instantiated to act as a pool of machines for use by students. If extra machines are needed, they can be quickly created. The snapshot feature would be useful for replaying exercises or reverting to known good confi gurations. Work has already been done on using VM approaches in training environments [3]: a simple GUI integrated with Grid tools would further this work, particularly for courses aimed at system administrators.

5 Related work

SODA [11] and Edge Services [8] support remote creation of VMs hosting application services; their interface is programmatic rather than visual. VM software providers such as VMWare and XenSource supply interfaces for managing VMs. These are not primarily designed for Grid use, and so lack Grid certificate authentication and integration with Grid middleware. Tregar describes a multi-VM build system constructed using Perl and VMWare [17]. His focus is on automating the creation of virtual machines rather on providing an interface to serve novice users.

6 Conclusion

The automated system we have presented makes it easy for all users of a testbed to quickly configure a Grid testing environment that suits their individual requirements. By storing standard configurations and filesystem images, the user can largely be shielded from the intricate details of Grid middleware and VM configuration. GridBuilder uses standard X509-based Grid certificates for authentication and is integrated with standard Grid middleware configuration tools used within the EGEE Grid.

There is scope for further work: the use of a centralised network repository for system images would enable multiple VM servers in a single testbed to act as a pool of hosts on which many VMs could be created. In our current prototype support for hosting multiple networks is implemented on routers outside GridBuilder; we would like to implement support for running an entire isolated testbed on a single machine. GridBuilder would isolate the hosted VMs, allowing them to be configured with arbitrary network addresses. On a more mundane level, support for configuring other Grid middleware such as Globus GT4 would expand the potential audience for the tool.

7 Acknowledgements

This work was supported by the Enabling Grids for E-Science project, funded by the European Union under contract number INFSO 508833.

References

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*. ACM, 2003.

[2] F. Bellard. QEMU CPU Emulator. `http://fabrice.bellard.free.fr/qemu/`, 2004.

[3] R. Berlich and M. Hardt. Grid in a box - virtualisation techniques in Grid training. Presented at EGEE conference, Athens, April 2005. Available via: `http://www.ep1.rub.de/~ruediger/pandoraAthens.pdf`.

[4] S. Burke, S. Campana, A. D. Peris, F. Donno, P. M. Lorenzo, R. Santinelli, and A. Sciaba. gLite 3.0 user guide. `http://edms.cern.ch/file/722398//gLite-3-UserGuide.pdf`, 2006.

[5] S. Childs, B. Coghlan, D. O'Callaghan, G. Quigley, and J. Walsh. Deployment of grid gateways using virtual machines. In P. M. Sloot, A. G. Hoekstra, T. Priol, A. Reinefeld, and M. Bubak, editors, *Advances in Grid Computing - EGC 2005*, LNCS3470, Amsterdam, The Netherlands, February 2005. Springer.

[6] S. Childs, B. Coghlan, D. O'Callaghan, G. Quigley, J. Walsh, and E. Kenny. A virtual testgrid or how to replicate a national grid. In *Proceedings of the EXPGRID workshop on Experimental Grid testbeds for the assessment of large-scale distributed applications and tools*, Paris, June 2006.

[7] 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.

[8] A. S. R. et al. An Edge Services Framework (ESF) for EGEE, LCG, and OSG. In *Proceedings of Computing in High Energy Physics*, Febrary 2006.

[9] L. Field and L. Poncet. LCG generic installation and configuration. `http://grid-deployment.web.cern.ch/grid-deployment/documentation/LCG2-M%anual-Install.pdf`, 2005.

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

[11] X. Jiang and D. Xu. SODA: A service-on-demand architecture for application service hosting utility platforms. In *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03)*, 2003.

[12] K. D. K. Keahey and I. Foster. From Sandbox to Playground: Dynamic Virtual Environments in the Grid. In *5th International Workshop in Grid Computing (Grid 2004)*, November 2004.

[13] R. G. Leiva, M. B. Lopez, G. C. Melia, B. C. Marco, L. Cons, P. Poznanski, A. Washbrook, E. Ferro, and A. Holt. Quattor: Tools and Techniques for the Configuration, Installation and Management of Large-Scale Grid Computing Fabrics. *Journal of Grid Computing*, 2(4), 2004.

[14] A. LeSuisse. `http://antony.lesuisse.org/qweb/trac/wiki/AjaxTerm`, 2006.

[15] A. Lewis. LVM HOWTO. `http://www.tldp.org/HOWTO/LVM-HOWTO`, October 2004.

[16] A. McNab. Grid-based access control for Unix environments, filesystems and web sites. In *Computing in High Energy and Nuclear Physics*, March 2003.

[17] S. Tregar. Perl, VMWare, & Virtual Solutions. *Dr. Dobb's Journal*, February 2005.