Global-scale service deployment in the XenoServer platform

Evangelos Kotsovinos, Tim Moreton, Ian Pratt, Russ Ross,
Keir Fraser, Steven Hand, Tim Harris[*]
University of Cambridge Computer Laboratory
15 J J Thomson Avenue, Cambridge, UK

Abstract

We are building the *XenoServer* platform for global computing, a public infrastructure capable of safely hosting untrusted distributed services on behalf of uncooperative paying clients. Service components execute on one or more XenoServers within resource-managed Virtual Machines (VMs) which provide resource isolation, protection, and allow the execution of arbitrary applications and services.

To assist the deployment of services on the platform, we provide an effective solution that allows users to fully customize the VMs to be launched by specifying the operating system kernel image and distribution file-system to be used. Moreover, we have implemented mechanisms for facilitating easy and efficient distribution of those kernel and file-system images; users build their VMs' configurations once and use the platform to efficiently launch VMs on large numbers of machines around the world.

Initial experiences with our deployment infrastructure demonstrate that the platform provides a practical substrate for public global computing; we show how a complex service running on the user's own customized Linux environment can be deployed to multiple XenoServers around the world in under 45 seconds.

1 Introduction

The XenoServer project [1] is building a public infrastructure for global-scale service deployment. XenoServers are machines that undertake the safe execution of potentially competing, untrusted services, in exchange for money. We have devised solutions to several research challenges in our previous work; allowing the execution of untrusted services is possible by performing reliable resource isolation on the servers [2]. Traceability of sponsors of malicious tasks is ensured by auditing service activity, and our infrastructure for resource pricing, accounting, billing, and charging provides support for uncooperative user communities [3]. Powerful higher-level tools to assist users in server discovery and selection have been developed [4].

This paper focuses on *service deployment*; this is the step where users, after having selected a number of XenoServers on which the service is to be deployed, proceed to contact the XenoServers to configure and start the virtual machines that will accommodate the service components, and to launch the service components themselves.

The XenoServer platform needs more than the conventional ad hoc means used to deploy services in the wide-area, due to the following challenging requirements:

- **Ease of deployment:** It is necessary that the cost of deploying large-scale distributed services on XenoServers is low, both in terms of money and effort. We envisage offering users mechanisms to "configure once, deploy anywhere"; after preparing their VM configurations, launching services on large numbers of servers should be trivial.

- **Performance:** As the XenoServer platform is designed to support deployment over short timescales, it is necessary that launching services on multiple servers around the world can be done quickly.

- **Efficiency:** To provide users full and flexible control of the configuration of their Virtual Machines, each new VM is specified from the ground up in terms of a file-system image and kernel. In a naive deployment model, this would incur transfers of several gigabytes to each selected XenoServer for each service deployment, and would raise the cost of deployment to potentially prohibitive heights.

- **Support for migration:** Services are likely to be location sensitive, meaning that service instances may need to be migrated as search tools determine better deployment positions within the network. It is necessary that the deployment architecture allows services to move around at a low cost.

[*]Microsoft Research, Cambridge, UK

- **Parallel deployment:** Since services may be widely replicated, the platform must support parallel deployment and allow reconfiguration of individual replicas.

In this paper, we propose a model for flexible and efficient service deployment, allowing dynamic migration of virtual machines. This model is described in Section 3. Section 4 describes the prototype implementation of our deployment infrastructure. In Section 5 we present initial experimental results aiming to determine the efficiency and flexibility of the proposed architecture.

2 Related Work

Existing distributed deployment infrastructures comprise deployment models that are often adequate for the needs of the environments they are designed to serve, but unsuitable for general-purpose global-scale service deployment.

The PlanetLab [5] project provides a distributed research testbed with geographically distributed servers for the research community. It offers only basic support for service distribution, requiring users to connect over `ssh` to each node individually to copy, configure, and control the custom service, a process that may be tedious when deploying to hundreds of nodes. More recently, the CoDeploy[1] service has been developed. This considerably eases the task of distributing experimental software to a set of PlanetLab nodes, and operates efficiently by using the CoDeen [6] CDN. It is not aimed at distributing operating system kernels or entire file-system images, however.

The same model, where users have to individually transfer data required for service deployment to each server involved, and configure the machines, is followed by a number of other service deployment infrastructures, such as Denali [7] and Grid computing projects [8, 9, 10]. Grid services are deployed using the APIs provided by the grid infrastructure which usually employs mechanisms such as GridFTP [11] for data distribution.

System imaging is a technique that enables archiving and copying disk images – usually containing operating system distributions and applications. Images can be used to clone servers by automating image deployment and configuration. Partition Image[2] generates disk images and uses domain-specific data compression techniques, while Frisbee [12] also employs local-area multicast for efficient distribution of images in local networks.

Imaging systems focus on the replication of entire disks' – and sometimes memory – contents to other machines in the local network for ease of configuration. Our system is different in that it aims at global-scale data distribution at deployment time and support for per-node configuration, parallel deployment and virtual machine migration.

VMatrix [13] follows a similar concept to that of disk imaging, facilitating the imaging of the run-time state of the machine along with files on the disk, and distributes such "virtual machines" on servers for easier configuration. The Internet Suspend/Resume project [14] allows users to capture and transfer the state of machines through the network. It targets the movement of a single virtual machine between two points, and does not address parallel deployment or per-node customization.

3 Deployment Model

We use *overlaying* techniques to allow users to maintain customized views of common distribution file-systems and kernel images.

We define a number of immutable *template images* for operating system kernels, and file-systems; these images are likely although not guaranteed to be persistently cached at each XenoServer. Customers describe a *tailored image* in terms of modifications to these templates, also called an *overlay*. This greatly reduces the amount of data that must be shipped to a XenoServer during deployment, reducing the setup time for a new virtual machine. This further enables dynamic replication or migration of services to proactively respond to offered load.

Although template images are stored locally at the XenoServer, client overlays are remotely accessed across the network. This extra level of indirection means that clients may configure their overlay independently of where their virtual machine will be instantiated; example customizations might include new start-up scripts, SSL certificates, or software packages. Since the overlay is remote, it may be shared between multiple virtual machines running on a set of XenoServers (e.g. to facilitate replicated server instances) and easily accessed by migrating services. Several layers of stacking are also supported, hence allowing per-service instance customizations and writeable directories as required. In most deployment scenarios no user will ever login to the virtual machine; the system will boot and start services automatically, perhaps writing log information to a remote file-system directory.

The configuration used in our implementation, shown in Figure 1, assumes a trusted (by XenoServers and clients) distributed storage service we call *XenoStore*. Since XenoStore is trusted, it is reasonable for the management virtual machine (MVM) – the privileged VM on each XenoServer that is responsible for launching other VMs – to mount parts of the clients' storage area. Hence the overlaying functionality can be provided within the MVM by a re-exporting copy-on-write NFS server; client virtual machines boot from NFS root over the machine-local virtual

[1] http://codeen.cs.princeton.edu/codeploy/
[2] http://www.partimage.org

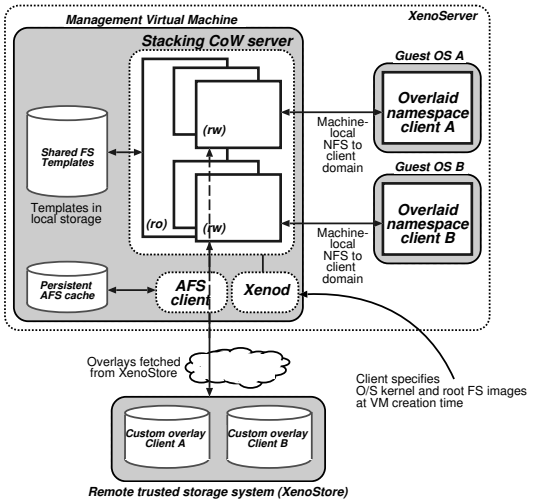

Figure 1: Deploying a Guest OS in the XenoStore model

network. NFS was chosen as it is supported by a wide range of Guest OSes.

XenoStore nodes are well-provisioned and well-connected servers, and so access latency should be low and data availability high. However it does require that users buy storage from XenoStore. We have also implemented – but do not discuss here due to lack of space – an alternative approach that does not require XenoStore, where clients implement the copy-on-write functionality within their virtual machine directly, and fetch overlays from their own, untrusted storage servers.

4 Implementation

Copy-on-write file server. A key component is a stacking copy-on-write file-system server, which overlays template images with one or more user-provided file-systems to construct the root file-system for a client VM. Our implementation interprets a `.mount` file in any directory to specify a list of file-systems to overlay at that subtree; this is presented as a unified namespace in which the order of the mounts specified determines which of two identically-named objects overrides the other, reminiscent of union directories in Plan 9 [15]. Modifications are written through to the first listed writeable file-system on a per file copy-on-write basis.

XenoStore. Our current implementation of XenoStore simply uses an existing distributed file-system to provide remote storage for users of the platform. We use the Andrew File System [16] since its persistent caching provides greatly increased performance relative to NFSv3 when used in the wide-area. To deal with security issues, we use an IPSEC VPN to connect to remote servers. Although we are working to replace AFS with a file-system designed precisely for our requirements, the current setup is certainly adequate to validate our approach and, as shown in Section 5, performs more than adequately well.

Deployment infrastructure. When using the XenoStore model (Figure 1), the user-provided deployment specification includes a URL identifying the overlaid file-system. A software component called `Xenod` parses the URL, and uses the scheme portion (e.g. `nfs://`, `afs://`) to determine the file-system type. It then mounts the remote overlay so that it is accessible by the MVM at a path chosen according to the new VM's identifier, and notifies the copy-on-write NFS server. This then exports that locally-accessible path as `/` such that it may only be mounted by the user's virtual machine; the link-local address Xen assigns to each virtual machine is unforgeable, and so is used for this purpose.

Subtrees in multiple file-systems can be overlaid at any point in a path, and hence mounting may be required on-demand. The stacking file-system server invokes `Xenod` to mount any such remote storage systems. This gives a clear separation between the manipulation of the overlaid namespace—performed by the stacking file-system—and the mounting of templates and remote file-systems.

By convention, `Xenod` uses the convention `template://` to name read-only operating system distribution templates. Immutable naming schemes provide a guarantee to a client using the template that the contents underneath it will not change. Mappings are also maintained from well-known names (e.g. `template://RedHat/9/current`) to these immutable identifiers, allowing 'default' distributions to be updated or have security patches applied. Choice of resolution of template identifier allows a client to specify the degree to which a service's file-system is subject to any template maintenance process.

5 Evaluation

In this section we evaluate the process of deploying a service both from scratch and by 'resuming' it from a previously suspended image. We focus on two applications illustrative of the types of service that might be common on the XenoServer platform — an Apache web server, and a Quake 3 game server. We breakdown the costs of the various deployment steps, using the XenoStore deployment model described in Section 3.

All experiments were performed between machines connected to our local Gigabit Ethernet. Our XenoServers were Dell 2650 machines configured with dual 2.4GHz Xeon processors, 2GB RAM, Broadcom Tigon 3 Gigabit Ethernet NICs, and Hitachi DK32EJ 146GB 10k RPM

Service	Overlay (KB)	Total FS (KB)	Proportion
Apache	23,695	2,318,937	1%
Quake3	533,671	2,828,913	18.8%

Table 1: Size of copy-on-write overlays.

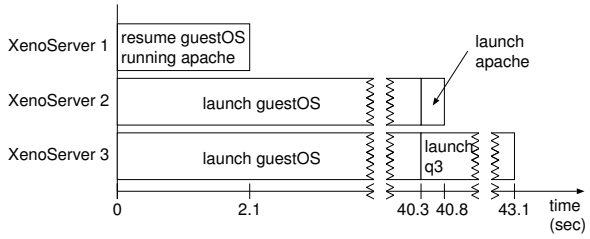

Figure 2: Service deployment timeline, showing individual operations and the time needed for each one.

SCSI disks. Overlays were stored on a dual processor 2.4GHz machine with 1GB RAM running a stock Andrew File System [16] server.

In order to measure wide area network effects under controlled conditions, the NISTNet [17] emulator was deployed on another machine configured as an IP gateway. We specified delay and bandwidth limits at 85ms and 108 KBytes/sec respectively, so that our configuration was illustrative of an arrangement in which the client and the XenoStore AFS server were in Cambridge, UK, the three XenoServers in New York. All RTTs were distributed normally with a standard deviation of 5% of the mean shown.

5.1 Overlay size

Before measuring the deployment process for the two services, we prepared overlays for them using the copy-on-write file-system server mounted loopback by a local NFS client over an immutable template. Table 1 shows that the total size of modified files required to support the services is a small fraction of the total file-system size. The impact on network traffic is discussed below.

5.2 Deployment timeline

Here we measure the time necessary for a complete service deployment. Each experiment was repeated 100 times, and measurements are taken from the UK client's perspective, including latency between the appropriate components.

For service deployment, the user contacts the XenoServers directly, which perform admission control, and assuming they accept the job, configure and instantiate virtual machines. The guest operating systems (Linux 2.4.26 in this example) boot and then deploy the target applications.

Our findings are shown in Figure 2. Users can deploy a new web server on several machines around the world in less than 41 seconds and a new quake3 server in just over 43 seconds. Most of the time taken is spent booting the guest OS to host the new service. The timeline also shows an example of using Xen's capability to suspend, migrate, and resume entire guest operating system images. By restoring an operating system image from a previously saved snapshot the application deployment time is reduced substantially, to just over 2 seconds in this example.

Components	New	Resumed
User - XenoServer	255 (37KB)	35 (6.5KB)
User - XenoStore	25193 (25.6MB)	none
XenoServer - XenoStore	1055 (731KB)	714 (720KB)

Table 2: Messages exchanged deploying a service from a new guest OS and a previously suspended one (total amount of data exchanged in brackets).

5.3 Network traffic

Table 2 shows the average network traffic generated, in terms of bandwidth and messages exchanged for the deployment of an Apache service, (a) using a completely new VM and (b) resuming a previously suspended VM.

The User–XenoStore figure accounts for the transfer of the Apache overlay from the AFS server. Since this will only be performed once for each service deployment – or more rarely, if overlays can be shared – we gain a significant saving on simpler models which require a transfer per virtual machine instantiation. This shows the efficiency of our service deployment mechanisms, and emphasizes the ease of service migration and redeployment using the suspend/resume mechanism.

6 Discussion

Our approach of using overlays allows the user to trade off the degree of customization they require against the ensuing impact on performance and real-world cost. As we have demonstrated in Section 5, the relative size of an overlay for realistic service deployments is generally small; however there is nothing to prevent a user from specifying an entirely bespoke file-system should they require it.

Using overlays also significantly eases management of replicated services. Since overlays can be constructed ahead of time and may be applied to more than one deployed instance, no administrative intervention is required when an instance migrates or is replicated. This is in contrast to PlanetLab [18] or Grids [19], for example, which require the user to actively transfer files to servers.

7 Conclusions and Future Work

In this paper, we have addressed challenges in wide-area data distribution so as to substantially lower the barrier to entry for deploying new services and applications on a global scale in the XenoServer platform. Our initial evaluation shows that the system can operate efficiently and provide rapid service deployment.

Using AFS over IPSEC in the prototype XenoStore has been a pragmatic solution that has given us useful experience. On the other hand, neither of these are particularly well-suited to our platform's environment, particularly in terms of their administrative model. We hope to incorporate ideas from SFS [20], Pond [21] and Pasta [22] to produce a more appropriate realization of XenoStore in the near future. In particular, we are developing a file-system which seamlessly combines the 'push' model of our distribution templates with the 'pull' model of demand caching.

We are in the process of preparing for a medium-scale XenoServer deployment, leasing a number of servers in commercial co-location facilities around the world. We welcome expressions of interest from members of the research community to act as beta testers.

References

[1] 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.

[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 *Proc. 19th ACM symposium on Operating Systems Principles (SOSP19)*, pages 164–177. ACM Press, 2003.

[3] Steven Hand, Timothy L Harris, Evangelos Kotsovinos, and Ian Pratt. Controlling the XenoServer Open Platform. In *Proc. 6th Int'l Conference on Open Architectures and Network Programming (OPENARCH)*, April 2003.

[4] David Spence and Tim Harris. Xenosearch: Distributed resource discovery in the xenoserver open platform. In *Proc. 12th IEEE symposium on High Performance Distributed Computing (HPDC-12)*, June 2003.

[5] L. Peterson, D. Culler, T. Anderson, and T. Roscoe. A blueprint for introducing disruptive technology into the internet. In *Proc. 1st Workshop on Hot Topics in Networks*, Princeton, NJ, October 2002.

[6] Limin Wang, Vivek Pai, and Larry Peterson. The Effectiveness of Request Redirecion on CDN Robustness. In *Proc. Fifth Symposium on Operating Systems Design and Implementation*, Boston, MA USA, December 2002.

[7] A. Whitaker, M. Shaw, and S. Gribble. Scale and performance in the Denali isolation kernel. In *Proc. 5th Symposium on Operating Systems Design and Implementation*, Boston, MA USA, December 2002.

[8] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *The Int'l Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, Summer 1997.

[9] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: The condor experience. *Concurrency and Computation: Practice and Experience*, 2004.

[10] I. Foster, D. Gannon, and H. Kishimoto. Open grid services architecture (OGSA). Technical report, Global Grid Forum, March 2004.

[11] W. Allcock, J. Bester, J. Bresnahan, S. Meder, P. Plaszczak, and S. Tuecke. GridFTP Protocol Specification, March 2003. Global Grid Forum – GridFTP Working Group Document.

[12] Mike Hibler, Leigh Stoller, Jay Lepreau, Robert Ricci, and Chad Barb. Fast, scalable disk imaging with frisbee. In *Proc. of the 2003 USENIX Annual Technical Conf.*, pages 283–296, San Antonio, TX, June 2003. USENIX Association.

[13] Amr Awadallah and Mendel Rosenblum. The vMatrix: A network of virtual machine monitors for dynamic content distribution. In *Proc. 7th Int'l Workshop on Web Content Caching and Distribution (WCW 2002)*, August 2002.

[14] Michael Kozuch and M. Satyanarayanan. Internet suspend/resume. In *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*, page 40. IEEE Computer Society, 2002.

[15] R. Pike, D. Presotto, K. Thompson, H. Trickey, and P. Winterbottom. The use of name spaces in Plan 9. In *Proc. 5th ACM SIGOPS European Workshop*, pages 72–76, 1992.

[16] Mirjana Spasojevic and M. Satyanarayanan. An empirical study of a wide-area distributed file system. *ACM Transactions on Computer Systems*, 14.

[17] N. Davies, G.S. Blair, K. Cheverst, and A. Friday. A Network Emulator to Support the Development of Adaptive Applications. In *2nd USENIX Symposium on Mobile and Location Independent Computing*, 1995.

[18] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak. Operating System Support for Planetary-Scale Network Services. In *Proc. 1st Symposium on Networked Systems Design and Implementation*, March 2004.

[19] Gridftp protocol specification, March 2003. Global Grid Forum Recommendation GFD.20.

[20] David Mazieres, Michael Kaminsky, M. Frans Kaashoek, and Emmett Witchel. Separating key management from file system security. In *Proc. 17th ACM Symposium on Operating Systems Principles*, pages 124–139, 1999.

[21] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz. Pond: The oceanstore prototype. In *Conference on File and Storage Technologies*, 2003.

[22] T. Moreton, I. Pratt, and T. Harris. Storage, Mutability and Naming in *Pasta*. In *Proc. Int'l Workshop on Peer-to-Peer Computing at Networking 2002, Pisa, Italy.*, May 2002.