Towards Peer-to-Peer Virtualized Service Hosting, Discovery and Delivery

David Bailey and Kevin Vella
University of Malta
Msida, Malta
Email: david@davidbailey.info, kevin.vella@um.edu.mt

***Abstract*—This paper introduces a peer-to-peer framework for providing, locating and consuming distributed services that are encapsulated within virtual machines. We believe that the decentralized nature of peer-to-peer networks acting in tandem with techniques such as live virtual machine migration and replication facilitate scalable and on-demand provision of services. Furthermore, the use of virtual machines eases the deployment of a wide range of legacy systems that may subsequently be exposed through the framework.**

To illustrate the feasibility of running distributed services within virtual machines, several Hadoop benchmarks are executed on a compute cluster running our framework, and their performance characteristics are evaluated. While I/O-intensive benchmarks suffer a penalty due to virtualization-related limitations in the prevailing I/O architecture, the performance of processor-bound benchmarks is virtually unaffected. Thus, the combination of peer-to-peer technology and virtualization merits serious consideration as a scalable and ubiquitous basis for distributed services.

***Keywords*-Virtualization, distributed systems, peer-to-peer computing, service-oriented computing**

I. Introduction

In recent years, data centre operations have experienced a shift in focus away from managing physical machines to managing virtual machines. Renewed exploration of this well-trodden path is arguably driven by virtualization's mantra of enhanced operational agility and ease of management, increased resource utilisation, improved fault isolation and reliability, and simplified integration of multiple legacy systems. Virtualization is also permeating the cluster and grid computing communities, and we believe it will feature at the heart of future desktop computers and possibly even advance a rethink of general purpose operating system architecture.

The performance hit commonly associated with virtualization has been partly addressed on commodity computers by recent modifications to the x86 architecture [1], with both AMD and Intel announcing specifications for integrating IOMMUs (Input/Output Memory Management Units) with upcoming architectures. While this largely resolves the issue of computational slow-down and simplifies hypervisor design, virtualized I/O performance will remain mostly below par until I/O devices are capable of holding direct and concurrent conversations with several virtual machines on the same host. This generally requires I/O devices to be aware of each individual virtual machine's memory regions and demultiplex transfers accordingly. We assume that this capability or a similar enabler will be commonplace in coming years, and that the commoditization of larger multi-core processors will reduce the frequency of expensive world-switches as different virtual machines are mapped to cores over space rather than time.

This paper introduces Xenos [2], a proof-of-concept implementation of a framework that enables the dynamic provision, discovery, consumption and management of software services hosted within distributed virtual machines. Xenos uses a decentralised peer-to-peer overlay network for advertising and locating service instances and factories. It also leverages techniques such as live virtual machine migration and replication to enhance operational agility and ease of management, and to lay the foundations for deploying fault-tolerant services. The primary objective is to shift the focus away from managing physical or virtual machines to managing software services.

This paper is organized as follows. Section II refers to some related work. Section III describes our proposed framework and the implemented prototype. Section IV presents an evaluation of the framework, and Section V discusses some topics for future investigation. We conclude in Section VI.

II. Related Work

The ideas presented here are influenced by the Xenoservers project [3], initiated by the creators of the Xen hypervisor. Xenoservers was designed to "build a public infrastructure for wide-area distributed computing" by hosting services within Xen virtual machines, while Xenosearch [4] locates Xenoservers using the Pastry peer-to-peer overlay network. A Xenoservers implementation is not generally available, hence our decision to build and conduct experiments with Xenos.

WOW [5] also uses a peer-to-peer overlay network to maintain self-organizing virtual links between virtual machines. IP connectivity is thus preserved across virtual machine migrations. However, WoW does not support the discovery of services on the overlay network.

Several other publications have focused on the use of peer-to-peer overlay networks to implement distributed resource indexing and discovery schemes in grid frameworks, such as [6], [7] and [8]. Wadge [9] investigates the use of peer groups to provide services in a grid, as well as transferring

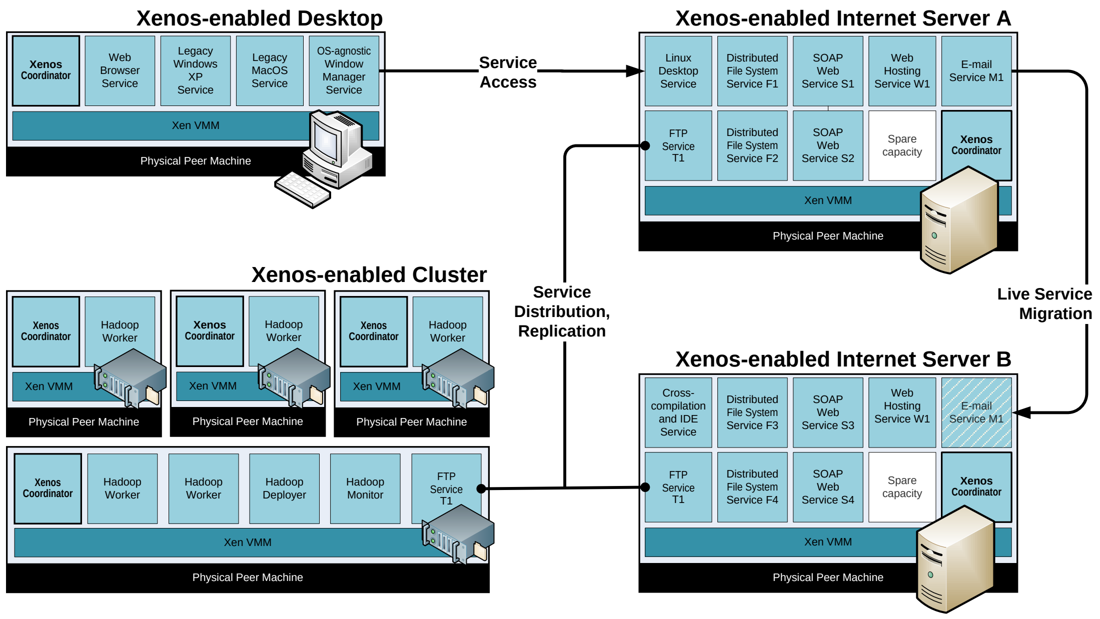


Figure 1. A selection of computing platforms running the Xenos framework and hosting several interacting services.

service code from one node to another for increased fault-tolerance and availability. This is achieved through the use of dynamically loadable non-virtualized plug-ins that can be shared by peers. SP2A [10] is a service-oriented peer-to-peer architecture which provides resource sharing in a non-virtualized grid.

The virtualized deployment of application software with specialized hardware requirements has also been explored in the literature, particularly in the context of high-performance and cluster computing: [11], [12] and [13].

III. The Xenos Framework

Xenos is built on top of Xen, a virtualization platform that has gained traction as a stable and mature virtualization solution, but any hypervisor with the appropriate hooks and programming interfaces will suffice in principle, including a hypothetical ROM-based hypervisor. The JXTA framework is currently used to maintain a peer-to-peer overlay network for service advertisement, discovery and, optionally, transport. However, we feel that a more specialized or expressive latter generation peer-to-peer framework would better fit our requirements.

A. Physiology

Figure 1 illustrates a scenario with different hardware platforms running Xenos and a variety of services. A compute cluster service enables users to dynamically create computation service instances, such as Hadoop map-reduce nodes. A pair of servers in a data centre offer web hosting, FTP, email and other services, all hosted within virtual machines and discoverable by users and other services across the Xenos cloud. Xenos also runs on a desktop computer, hosting several light-weight services (virtualized Google ChromeOS, for instance).

B. Architecture

Each Xenos-enabled physical machine runs the Xen hypervisor using a paravirtualized Linux kernel in Domain 0, which is a privileged domain capable of controlling the guest domains that will host services on the same physical machine. The Xenos coordinator is a Java application that executes in Domain 0 whose primary function is to incorporate the physical machine into Xenos's peer-to-peer overlay network and advertise services running on that physical machine. Services running within guest domains do not normally join the overlay network directly, but are registered with the coordinator in Domain 0 which acts as a 'notice board' for all local services. Xenos provides an XML-RPC programming interface for users and services to discover, locate and manage services.

Service delivery itself may be accomplished without the involvement of Xenos, and is not restricted to any particular network protocol or address space. However, the direct use of network protocols beneath layer three (for example, Ethernet) would oblige communicating services to share a physical network or a physical machine. Figure 2 illustrates the architecture of a single physical machine in the framework.

In order to accommodate multiple instances of the same service and service migration, each service type has a template associated with it that enables the automatic configuration of new service instances and their Xen domains. When replicating a service or creating a new service instance, a new copy of the relevant template is used. Service templates will automatically replicate on other Xenos hosts as required so that service instances can be spawned anywhere on the Xenos cloud. Migration of service instances makes use of Xen's virtual machine migration mechanism with a

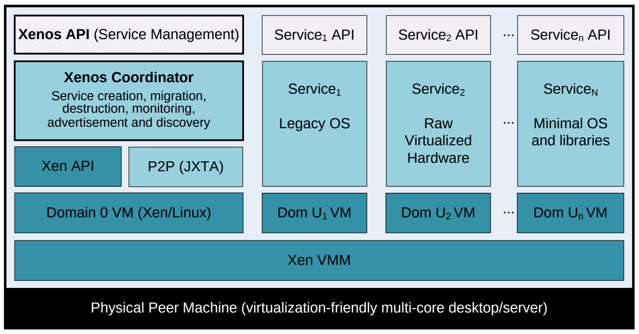


Figure 2. A Xenos-enabled physical machine.

slight modification to transfer virtual machine disk images along with the virtual machine configuration. Our current implementation inherits a Xen restriction limiting live virtual machine migration to the local area network, though this may be overcome as discussed in Section V.

IV. Performance Analysis

A series of preliminary tests were conducted in order to assess the viability of our approach. The test cases all involve deploying multiple instances of a Hadoop map-reduce wrapper service using a separate distributed coordination service. We aim to explore three principal avenues, namely (1) the automatic and dynamic deployment of the Hadoop service to Xenos hosts and the migration of the master Hadoop node from a failing physical machine; (2) the performance of file I/O within virtual machines, which is crucial for services with large-volume data processing requirements (this is particularly relevant since Xenos requires virtual machine images to exist in files rather than as physical disk partitions); and (3) the performance of a series of virtualized Hadoop map-reduce processing jobs.

A similar evaluation of running the Hadoop map-reduce framework within a virtualized cluster is carried out by Ibrahim *et al.* [14]. They argue that a virtual machine-based Hadoop cluster can offer compensating benefits that overshadow the potential performance hit, such as improved resource utilization, reliability, ease of management and deployment, and the ability to customize the guest operating systems that host Hadoop to increase performance without disrupting the cluster's configuration.

A. Map-Reduce and Hadoop

In our experiments we used the HDFS (Hadoop Distributed File System) and MapReduce components of the Apache Hadoop framework. The map-reduce programming model, introduced by Dean *et al.* [15], is aimed at processing large amounts of data in a distributed fashion on clusters. HDFS is a distributed file system suitable for storing large data sets for applications with heavy data processing, such as typical map-reduce jobs. The Hadoop map-reduce implementation involves a master node that runs a single *JobTracker*, which accepts jobs submitted by the user, schedules the job across worker nodes by assigning map or reduce tasks to them, monitors these tasks and re-executes failed ones. Each worker node runs a single *TaskTracker* which is responsible for executing the tasks assigned to it by the job tracker on the master node.

B. Deploying Hadoop Services

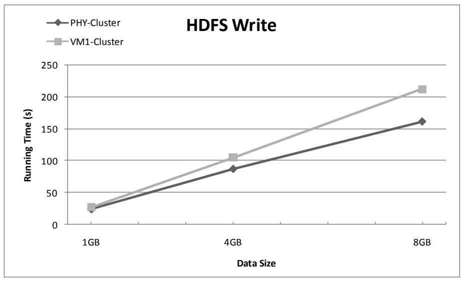
Each Hadoop map-reduce node needs to be configured with specific settings, such as the host name, host certificates and HDFS and map-reduce settings that are common throughout the cluster. Setting up a non-virtualized environment usually involves manually configuring a single node, then cloning the hard disk to the rest of the cluster, either manually or via shell scripts and *rsync*.

Our approach is to encapsulate a pre-configured Hadoop installation inside a virtual machine and automatically distribute it across the Xenos network as a service. To facilitate the distribution, we developed a Java/JXTA application to connect to the Xenos cloud, and used the Xenos programming interface to deploy a Hadoop slave worker service. One of the hosts on the cluster, which we refer to as the master host, is configured with a template of the Hadoop slave service as well as an instance of the Hadoop master service, from where we issue commands to deploy services and execute Hadoop jobs.

C. Evaluation Platform and Results

The evaluation platform was a thirteen-host cluster, connected over a 1GB/s Ethernet connection through a D-Link DGS-1224T switch. Each physical machine in the cluster has an Intel Core 2 Duo E7200 CPU, with 3MB of L2 cache clocked at 2.53GHz, 2GB of DDR2 RAM, and a 500GB SATA2 hard disk. In all of our tests, the virtual machine that we use as the Hadoop slave template which is replicated is configured with a 10GB disk image, a 1GB swap image, the *vmlinuz-2.6.24-27-xen* kernel, one VCPU (virtual CPU), 384MB of RAM and a DHCP client. Domain 0 is set to use 512MB of memory, leaving the rest to be allocated to service-hosting virtual machines, and has no restrictions on the number of physical CPUs it can use. One of the cluster hosts is dedicated to hosting the Hadoop slave template and the master service instance, and configured so that no slave services are replicated on it. No optimizations to Hadoop or any other software component were made to suit this particular cluster. In all results, PHY-Cluster refers to a Hadoop cluster on native Linux, while VM1-Cluster, VM2-Cluster and VM4-Cluster refer to Xenos-enabled virtualized clusters with one, two and four virtualized Hadoop slave services deployed per physical host respectively.

1) Replication and Migration of Hadoop Service Templates and Services: The unoptimized replication process took around 45 minutes to deploy a template and a single slave service instance to each of the twelve remaining cluster hosts, which included a network transfer of 132GB as well



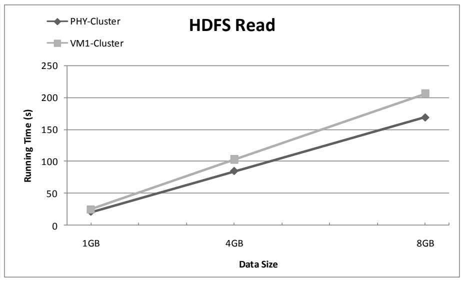
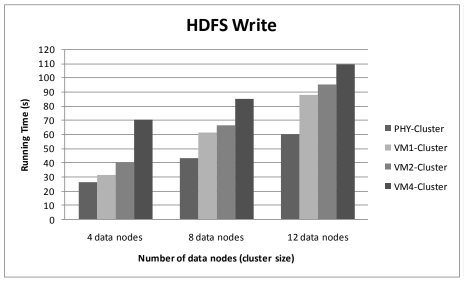


Figure 3. PHY-Cluster *vs* VM1-Cluster with varying data sizes.



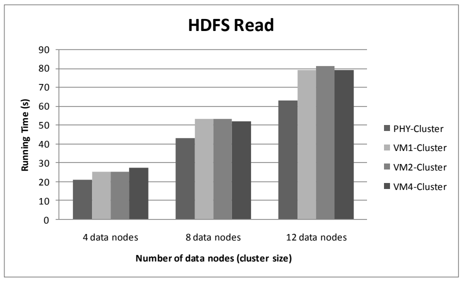


Figure 4. PHY-Cluster *vs* VM clusters with varying data nodes (cluster size) and virtual machines per physical machine.

as another 132GB in local data copying; this translates to a network throughput of around 40MB/s and a disk throughput of around 25MBs/s. Since the process mostly involves transferring domain files over the network or copying them locally, its performance depends on the hardware platform that the services are being deployed on, as well as the size of the domains that contain the service. Once the required templates have been automatically deployed and replicated throughout the cluster, activating existing services takes a tiny fraction of this time. Additionally, the Hadoop master service was successfully migrated between hosts to simulate a physical host that needs to be dynamically repurposed.

2) HDFS Performance: As shown in Figure 3, reading and writing operations on virtualized HDFS suffered a performance drop when compared to a non-virtualized cluster configuration. For small data transfers and clusters, the gap is negligible, but increases with larger data sets. However, as shown in Figure 4, increasing the number of virtualized services per physical host did not cause the read performance of HDFS to deteriorate. Ibrahim *et al.* also make this observation in one of their tests, indicating that the write performance gap increased markedly but it increased only slightly when reading.

3) Hadoop Benchmarks: Figure 5 indicates that increasing the number of computation nodes by adding more virtualized service instances on each physical machine benefits certain processor-intensive Hadoop jobs. In this case, the PiEstimator benchmark performed significantly better when more computing nodes were available. However, jobs that are I/O-intensive and that deal with large data sets suffered a performance hit due to degraded HDFS performance; this was evident in the Wordcount and Sort benchmarks

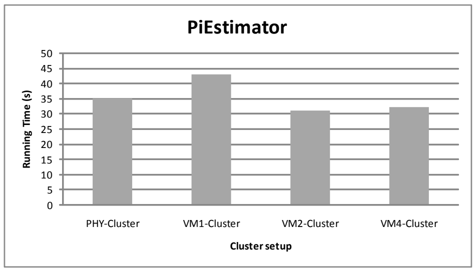


Figure 5. PiEstimator execution on PHY-Cluster and VM clusters with varying virtual machines per physical machine.

as illustrated in Figure 6 and Figure 7. In the Wordcount benchmark, Ibrahim *et al.* fared better on their virtualized clusters with 2 and 4 VMs per physical host than their physical cluster; however each host in their evaluation was equipped with 8 cores, so their CPU core to VCPU ratio was always 1 or greater. Our Sort benchmark results are similar to Ibrahim *et al.*'s: we also observed that once the reducer tasks start executing, the entire job slows down considerably.

As discussed in Section I, we expect future improvements in virtualization technology to further minimize the gap between native and virtualized I/O performance, thus strengthening the case for deploying Xenos and other such platforms.

V. Topics For Further Investigation

With some effort, Xenos can fill the role of a test-bed to facilitate experimentation with a variety of emerging issues in distributed virtualized services, some of which are briefly discussed here.

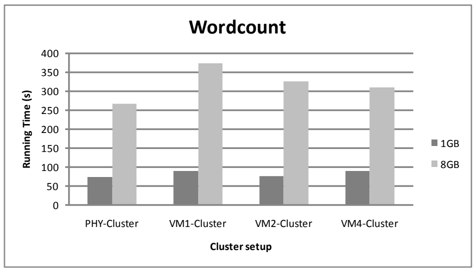


Figure 6. Wordcount execution on PHY-Cluster and VM clusters with varying data input size and virtual machines per physical machine.

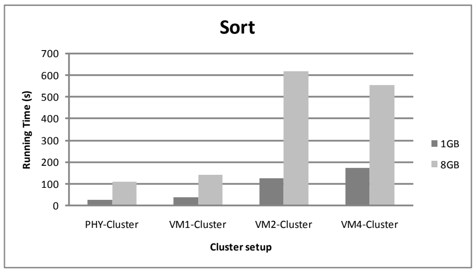


Figure 7. Sort execution on PHY-Cluster and VM clusters with varying data input size and virtual machines per physical machine.

A. A Library of Essential Services

The core functionality provided by the Xenos framework can be further extended and abstracted away through additional services. Examples include service wrappers for load-balancing and fault-tolerance (virtual machine checkpointing is invisible to the service(s) hosted within), virtual machine pooling and replication, service deployers such as the Hadoop deployer discussed previously, platform emulators, legacy services supporting a range of operating systems, and a Xenos-UDDI adapter that can be used to search for Xenos services via UDDI (Universal Description Discovery and Integration). Xenos does not impose a single method for actual service delivery, thus web services, Sun RPC, and even services using raw Ethernet may be advertised.

B. Seamless Wide-Area Service Migration

The issue of live virtual machine migration over WANs has been addressed by several authors and a number of prototypes are available. Travostino *et al.* [16] approach the problem of preserving TCP connections by creating dynamic IP tunnels and assigning a fixed IP address to each virtual machine, which communicates with clients via a virtual gateway interface that is set up by Xen. After migration, a virtual machine retains its address, and the IP tunnels are configured accordingly to preserve network routes – this is completely transparent to TCP or any other higher level protocol. Bradford *et al.* [17] combine the IP tunneling approach with Dynamic DNS to address the problem of preserving network connections. More importantly, the authors also implement a pre-copy approach for transferring the disk image attached to a virtual machine, using a mechanism similar to that used by Xen when live migrating the state of a virtual machine. This greatly minimizes downtime even if the actual migration takes long owing to poor network performance. Harney *et al.* [18] suggest using the mobility features in the IPv6 protocol to preserve network communication sessions, an approach that is viable in the long-term.

C. Alternative Transport Methods For Service Delivery

Applications featuring fine grained concurrency spanning across virtual and physical machines stand to gain from inter-virtual machine communication path optimizations such as shared memory communication for services residing on the same physical machine, and hypervisor-bypass network communication for distributed services. In both instances, the secure initialization of each communication path would be delegated to Xenos, allowing the data to move directly between the participating virtual machines and virtualization-enabled I/O devices. In some cases, an I/O could be permanently and exclusively bound to a specific service for low-latency dedicated access.

D. Security, Authentication and Service Provisioning

A number of underlying mechanisms could be inherited from the Xen hypervisor and the JXTA peer-to-peer framework or their respective alternatives. To our benefit, JXTA provides several security and authentication features, as discussed by Yeager *et al.* [19]; these include TLS (Transport Layer Security), and support for centralized and distributed certification authorities. Xen provides a basis for automated accounting and billing services that track service consumption as well as physical resource use. However, Xenos should at least provide unified and distributed user, service and hierarchical service group authentication and permission mechanisms, a non-trivial undertaking in itself.

E. The Operating System-Agnostic Operating System

Software architectures in the vein of Xenos could fit the role of a distributed microkernel in a virtualization-embracing operating system that consists of interacting light-weight services hosted within virtual machines, including a multi-core thread scheduler, file systems (a stripped down Linux kernel), and device drivers. Each operating system service would run within its own light-weight Xen domain and expose itself through Xenos services (reminiscent of system calls). Xenos services would also host legacy operating systems and applications, presented to users through an operating system-agnostic window manager hosted in a separate virtual machine. Applications with particular resource requirements or requiring isolation, such as computer games or web browsers, may easily be hosted in their own virtual machines, supported by a minimal application-specific kernel or library or even executing on ‘bare virtualized metal’. Xen, and virtual machine monitors in general, have been described as “microkernels done right” [20], although others have argued that the drawbacks that muted the adoption of microkernels [21] still apply.

VI. Conclusion

This paper briefly investigates an approach to building distributed middleware. The Xenos framework extends well-established solutions for virtualization hypervisors and peer-to-peer overlay networks to deliver the beginnings of a fully decentralized solution for virtualized service hosting, discovery and delivery. We expect forthcoming hardware support for virtualization to further reduce the gap between virtualized and native I/O performance pinpointed in our results, while simplifying and possibly commoditizing hypervisors. This will further consolidate the virtual machine's position as a viable alternative for hosting both computation- and I/O-intensive tasks. The combination of peer-to-peer technology and virtualization merits serious consideration as a basis for resilient distributed services.

References

[1] P. Willmann, S. Rixner, and A. L. Cox, "Protection strategies for direct access to virtualized I/O devices," in *ATC'08: USENIX 2008 Annual Technical Conference on Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2008, pp. 15–28.

[2] D. Bailey, "Xenos: A service-oriented peer-to-peer framework for paravirtualized domains," Master's thesis, University of Malta, Submitted 2010.

[3] K. A. Fraser, S. M. Hand, T. L. Harris, I. M. Leslie, and I. A. Pratt, "The XenoServer computing infrastructure," University of Cambridge Computer Laboratory, Tech. Rep., 2003.

[4] D. Spence and T. Harris, "XenoSearch: Distributed resource discovery in the XenoServer open platform," in *HPDC '03: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*. Washington, DC, USA: IEEE Computer Society, 2003, p. 216.

[5] A. Ganguly, A. Agrawal, P. O. Boykin, and R. Figueiredo, "WOW: Self-organizing wide area overlay networks of virtual workstations," in *In Proc. of the 15th International Symposium on High-Performance Distributed Computing (HPDC-15*, 2006, pp. 30–41.

[6] V. March, Y. M. Teo, and X. Wang, "DGRID: a DHT-based resource indexing and discovery scheme for computational grids," in *ACSW '07: Proceedings of the fifth Australasian symposium on ACSW frontiers*. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2007, pp. 41–48.

[7] Q. Xia, R. Yang, W. Wang, and D. Yang, "Fully decentralized DHT based approach to grid service discovery using overlay networks," *Computer and Information Technology, International Conference on*, pp. 1140–1145, 2005.

[8] D. Talia, P. Trunfio, J. Zeng, and M. Hgqvist, "A DHT-based peer-to-peer framework for resource discovery in grids," Institute on System Architecture, CoreGRID - Network of Excellence, Tech. Rep. TR-0048, June 2006.

[9] W. Wadge, "Providing a grid-like experience in a P2P environment," Master's thesis, University of Malta, 2007.

[10] M. Amoretti, F. Zanichelli, and G. Conte, "SP2A: a service-oriented framework for P2P-based grids," in *MGC '05: Proceedings of the 3rd international workshop on Middleware for grid computing*. New York, NY, USA: ACM, 2005, pp. 1–6.

[11] S. Thibault and T. Deegan, "Improving performance by embedding HPC applications in lightweight Xen domains," in *HPCVirt '08: Proceedings of the 2nd workshop on System-level virtualization for high performance computing*. New York, NY, USA: ACM, 2008, pp. 9–15.

[12] M. J. Anderson, M. Moffie, and C. I. Dalton, "Towards trustworthy virtualisation environments: Xen library OS security service infrastructure," Hewlett-Packard Laboratories, Tech. Rep. HPL-2007-69, April 2007. [Online]. Available: http://www.hpl.hp.com/techreports/2007/HPL-2007-69.pdf

[13] E. Van Hensbergen, "P.R.O.S.E.: Partitioned reliable operating system environment," *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 2, pp. 12–15, 2006.

[14] S. Ibrahim, H. Jin, L. Lu, L. Qi, S. Wu, and X. Shi, "Evaluating MapReduce on virtual machines: The Hadoop case," in *CloudCom*, 2009, pp. 519–528.

[15] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[16] F. Travostino, P. Daspit, L. Gommans, C. Jog, C. de Laat, J. Mambretti, I. Monga, B. van Oudenaarde, S. Raghunath, and P. Y. Wang, "Seamless live migration of virtual machines over the MAN/WAN," *Future Gener. Comput. Syst.*, vol. 22, no. 8, pp. 901–907, 2006.

[17] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg, "Live wide-area migration of virtual machines including local persistent state," in *VEE '07: Proceedings of the 3rd international conference on Virtual execution environments*. New York, NY, USA: ACM, 2007, pp. 169–179.

[18] E. Harney, S. Goasguen, J. Martin, M. Murphy, and M. Westall, "The efficacy of live virtual machine migrations over the internet," in *VTDC '07: Proceedings of the 3rd international workshop on Virtualization technology in distributed computing*. New York, NY, USA: ACM, 2007, pp. 1–7.

[19] W. Yeager and J. Williams, "Secure peer-to-peer networking: The JXTA example," *IT Professional*, vol. 4, pp. 53–57, 2002.

[20] S. Hand, A. Warfield, K. Fraser, E. Kotsovinos, and D. Magenheimer, "Are virtual machine monitors microkernels done right?" in *HOTOS'05: Proceedings of the 10th conference on Hot Topics in Operating Systems*. Berkeley, CA, USA: USENIX Association, 2005.

[21] G. Heiser, V. Uhlig, and J. LeVasseur, "Are virtual-machine monitors microkernels done right?" *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, pp. 95–99, 2006.