DESIGN SPACE EXPLORATION OF VIRTUAL MACHINE APPLIANCES FOR WIDE-AREA DISTRIBUTED COMPUTING

By

DAVID ISAAC WOLINSKY

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2007

I dedicate this to Donna; I deeply appreciate her understanding and pray that the next time I am not so overwhelmed.

ACKNOWLEDGMENTS

I have many to thank for successfully getting this far. By far the most important is Donna, who continually encourages me to pursue my passions. I deeply appreciate Professor Figueiredo for giving me my first graduate assistantship and following me through to graduation. Further, I thank both Professor Figueiredo and Boykin for their hard work and dedication toward my research and related research. Also, I thank Professor Lam for taking a chance on me what seems like an eternity ago. Also, my parents who have always encouraged me even though as time goes on, they understand less and less of what I am doing. Finally, I know deep in my heart that without a loving and forgiving God, that none of this would have been possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

DESIGN SPACE EXPLORATION OF VIRTUAL MACHINE APPLIANCES FOR WIDE-AREA DISTRIBUTED COMPUTING

By

David Isaac Wolinsky

May 2007

Chair: Renato J. Figueiredo
Major: Electrical and Computer Engineering

Virtual machines (VMs) provide an ideal way to package complete systems in a file and deploy complete application environments without hindering pre-existing software on a computer. By using VMs, the ability to develop, deploy, and manage distributed systems has been greatly improved. This paper explores the design space of VM-based sandboxes where the following techniques that facilitate the usability of secure nodes for grid computing: grid schedulers, DHCP-based virtual IP address allocation on virtual LANs, self-configuring virtual networks supporting peer-to-peer NAT traversal, stacked file systems, IPsec-based host authentication and end-to-end encryption of communication channels, and user interfaces. Experiments with implementations of single-image VM sandboxes, which incorporate the above features and are easily deployable on hosted I/O VMMs, show execution time overheads of 10.6% or less for a batch-oriented CPU-intensive benchmark.

CHAPTER 1
INTRODUCTION

1.1 Problem Statement

International Business Machines Corporation (IBM) proposes the idea of autonomic computing [1], which states that systems should be self-configuring, self-healing, self-optimizing, and self-protecting. This relates to many issues existing in grid computing [2], such as the deployment, maintenance, and accessibility of grid resources. Deployment of grid resources focuses primarily on the complexity of the software stack and its dependencies. Maintenance involves the minimum number of tasks required by an administrator in order to keep a well running grid system. The ability of users to interact with grid resources depends on how accessible the grid resource is. The problem is "What are the requirements for the grid software stack and how are they best met?"

The process of setting up a computing grid can be very detailed. Grid software tends to have many different complex configurations with relatively few plug and play systems; furthermore, many grid software packages still require other dependencies which are not included in the core software package. The issue can be complicated by requiring a specific or limited set of hardware configurations. Often time, grid software requires dedication of the underlying hardware leading to system utilization inefficiencies.

Though developers ideally want a grid system which never needs any updates in the software stack, often times issues are discovered after release or new, desirable features are made available. The issue of making these updates available to the system often times can be a complex process at a minimum requiring that the administrator at each grid node download the update, apply it to the system, and then tweak each system accordingly. There are a few systems which have automatic updating features and often times if they do they require super user level services which may interfere with using the hardware resource for other purposes. If an update breaks pre-existing software or configurations, this can lead to an even larger nightmare.

Users prefer common interfaces and few are willing to learn the complexities of text-based grid interfaces, that is, they are most comfortable with a graphical ”point and click” interface where only user level configuration data is visible and all system level configuration data is transparent to them. Even if a graphical user interface (GUI) was available, each different application would need its own unique interface and most GUIs are complex to program, which limits the amount of applications developers or administrators can make available via a ”point and click” system. In an age of frustrated informational technologists (IT) professionals and computer users, people are less likely to install foreign software on their computers, whether it be an ssh, X11, Virtual Network Computing (VNC), or grid system specific client.

1.2 Defining the Solution

The solution proposed in this thesis is called Grid Appliance and deals with the above issues while still limiting any compromise in terms of software or hardware configuration and with limited performance overhead. At the heart of Grid Appliance lies virtualization primarily through packaged virtual machines (VMs). As suggested by previous research [2], the Grid Appliance takes advantage of such technologies such as the Internet, distributed computing, and peer-to-peer networking.

The benefits of using virtual machines in grid computing [3] “include security, isolation, customization, legacy support, and resource control” with very limited overhead in processor intensive applications. Grid Appliance has been designed to run on the two most popular virtual machine technologies VMware [4] and Xen [5] and is in the process of working on VMs such as VirtualBox[6], Parallels [7], Qemu [8], and Kernel-based Virtual Machine (KVM) [9].

The most important use of virtualization in the Grid Appliance is encapsulation of not only virtualized disks but also allowing a full system to execute inside another in a non-obstrusive way. With respect to virtual disks VMs allow for the creation of file systems which are placed into a single portable file. With this capability, all the software

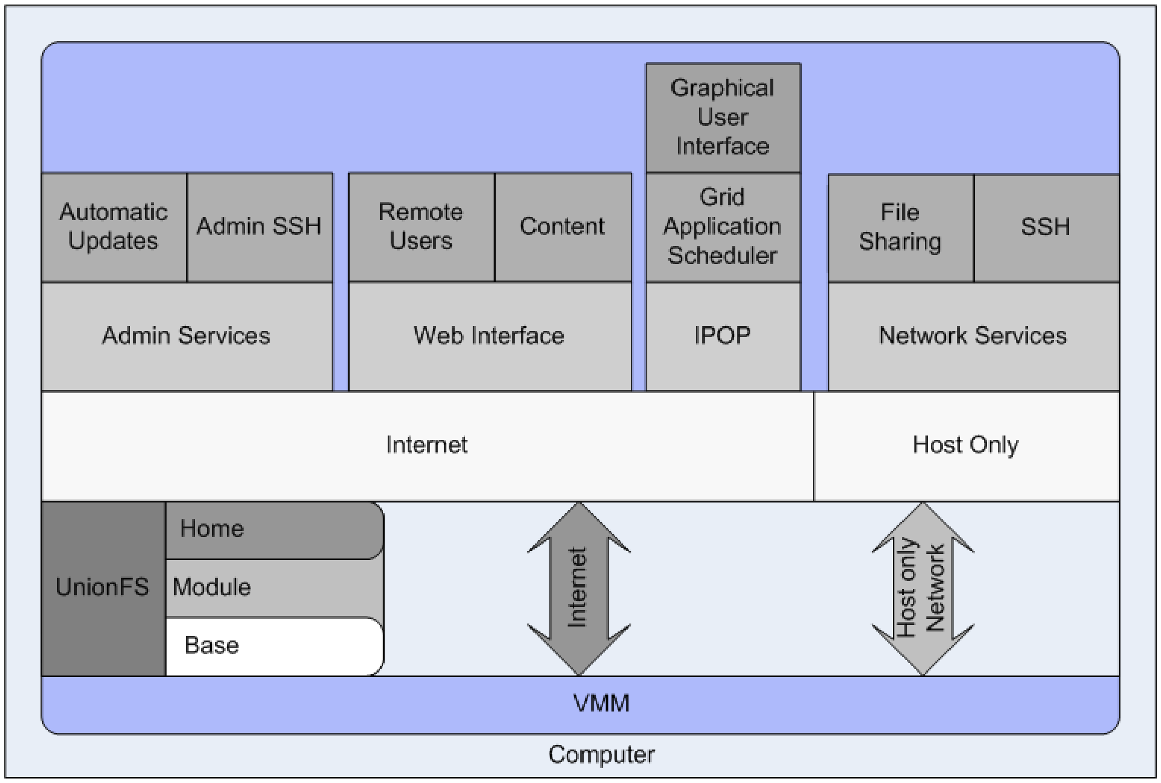


Figure 1-1. High-level architectural overview of the Grid Appliance.

.

used in the system is installed onto the virtual disk and since the software runs in an abstracted hardware no reconfiguration is required. The key software features of the Grid Appliance fall in these catogories (Figure 1-1): system services (a), miscellaneous services (b), administrative services (c), web interface (d), and network services (e).

System services are varied and include portable file systems via UnionFS and virtual network through IPOP (IP over Peer-to-Peer). Miscellaneous services contain software which can easily be interchanged such as the grid scheduler and local user interface library. Administrative services include automatic updates and administrative ssh. Web interface provides the capability for publishing of content as well as user access to the system. Users can access the command line and their files through the network services.

Some of this material has been presented before [10], which is the work of the same author. While this presents many new concepts and a wider breadth of information, it is closely related to the original paper. Due to this, this paper will only be referenced here, acknowlding that it was the premise for further and future investigation.

1.3 Thesis Outline

This thesis is outlined as follows: Chapter 2 reviews background projects, Chapter 3 discusses the construction of the Grid Appliance, Chapter 4 overviews related work, Chapter 5 validates and tests the performance of the Grid Appliance, and finally in Chapter 6 discusses the current deployment, concludes this thesis, and provides light on future work.

CHAPTER 2
BACKGROUND

The primary features of the Grid Appliance are virtual computing, grid computing, and virtual networking. This chapter reviews projects in those fields and presents arguments as to why a specific project was chosen over others for inclusion in the Grid Appliance.

2.1 Grid Computing

The Grid Appliance provides the ability to create wide area distributed systems with the focus on high throughput computing [11], in other words, the ability to provide a large amount of processing power over a large period of time. This is because wide area systems tend to not provide the best environment for parallel applications due to high latency; however, they are ideal for sequential applications.

The need for grid computing is there, users want to be able to submit many tasks in a secure environment with the knowledge that they will be processed in due time. This creates many obstacles focused primarily on users' ability to access the resources and ensuring there are enough resources available given the amount of users in the system. Most modern systems use a central manager scheme, which all users connect and submit their jobs to the cluster from these managers. The pitfall of this design is that it creates a single point of failure, where if the manager or managers go down, access to the system is denied. Furthermore, the effort associated with maintaining the system by IT personnel can be costly.

Adding new resources and managing existing resource is complex, error-prone, and costly; this happens to be the bulk cost of a system. Systems like PlanetLab [12] have reduced the complexity of adding new resources by providing a CD image and a configuration file that can be copied to a floppy drive or a USB device; however, this system still requires a central manager in terms of person and computing. The main computer provides the only way users can gain access to the individual nodes; furthermore, this central system also contains the main system image which all nodes must download before they can be used. Administrators are required to allow new users to access resources, initiate the process of resource adding, and to fix configuration bugs in existing resources.

The cluster computing software stack consists primarily of job managers such as PBS and its forks [13–15], Condor [16–18], Sun's Grid Engine [19], Load Sharing Facility [20], and RES. This software requires a central manager to which all worker nodes must have direct communication. In order to submit to the nodes, a user must have access to a machine which can communicate to all the worker nodes, where both machines are in the same address space. This removes the ability for a default configured Condor or PBS system to be able to talk from one private network to another private network. This is an issue where universities may want to set up a shared distributed computing grid but are unwilling to give each machine in the pool a public IP address due to the lack of public IP addresses or security issues. The solution proposed for this problem is virtual networking, which is discussed further in chapter 3.5.

The basic needs for the job manager [21, 22] in Grid Appliances case are

1. ability to handle hundreds of nodes and more jobs than workers;
2. allows any node that can execute jobs also submit;
3. the project is Open source and a free full version;
4. handles system and network issues well;
5. supports user-level check pointing;
6. provides for shared cycles from; desktops[1] ;
7. ability to prevent rogue nodes from submitting jobs.

There are three products based upon PBS (Portable Batch System), OpenPBS [14], PBS Pro [13], and Torque [15]. OpenPBS is an older version of PBS that was released to the open source community. According to Altair, it is not well suited for systems that want to run more than 100 jobs or more than 32 nodes. Altair's flagship job manager, PBS Pro, is closed source and is only free to the academic community but requires cash

[1] That is, if a machine is taken over by the local user, the job will suspend and restart later or migrate to an available machine.

backed license otherwise. TORQUE is a fork of the OpenPBS. TORQUE and PBS Pro have a similar feature set and are capable of handling thousands of nodes and more jobs than workers also with the support of interactive jobs. PBS and its derivatives were designed to run on dedicated resources and do not support and thus does not provide for point 6 mentioned above. Also there is only kernel-level support for check pointing.

Sun's Grid Engine has a similar feature set of PBS, except that it supports many more operating systems and allows for the execution parallel. Another bonus for the grid engine is that it supports user-level check pointing and handles point 6 fine. The Grid Engine comes in both a free and a support backed version. The source is available from the Internet. The packaging is robust and supported on many operating systems.

Condor is meant for shared resources, where a job will suspend if the computer is accessed by a local user; furthermore, Condor is free and open source, allowing for changes to Condor. Condor is the only scheduler out of these three to have a strong presence in academia, largely in part because it is still actively being developed by the University of Wisconsin. The Grid Appliance's primary users are expected to be academia, thus Condor being developed by a university and the other listed reasons, Condor was selected as the default scheduler in the Grid Appliance. The faculty related to the development of the Grid Appliance also had strong ties and background to Condor, thus using it allowed for less overhead in learning how to configure and use the system.

The last two mentioned Load Sharing Facility and RES are both closed source and require cash back licenses to use which goes against the principles for the Grid Appliance. While Condor was selected as the default manager for now, there was significant interest in the features that Grid Engine provided that Condor did not and will remain a possibility if those features are needed in the future.

Thanks to the Globus Alliance, pools can be shared in a secure fashion. Their sofware, Globus Toolkit [23], allows for the ability to build and unite grid applications and systems. For example, the schedulers Condor and PBS have released tools that allow external connection to their pools via Condor-G and PBS-Globus respectively. In fact, the current deployment of the Grid Appliance allows access to the condor pool via Condor-G.

Imagine a system where a user installs a light weight, non-intrusive application that connects directly to a distributed cluster running on homogeneous systems. This application requires no user input besides pressing the "virtual power on" button and updates are invisible to the user. The software requires no additional network cards besides an active connection to the Internet. If the user ever breaks the application or it has internal strife, all the user has to do is restart the application; often times however, the application will be able to recover on its own. This application is a "black box", the user's system has no idea what's running inside and nothing outside the system affect it and reciprocally nothing inside of it can affect the system. System mangers can turn on the software and walk away, while users' are given multiple different levels of entry depending on their needs and skill sets, such as web interface, console access, and direct file system access. This is all available in the form of the Grid Appliance. The remaining chapters discuss the merits of components of the Grid Appliance and their related works.

2.2 Virtualization

Back in the late 1960s and early 1970s, IBM led research into the use of virtual computing to time multiplex fully complete, isolated systems to individual users. The system was called CP/CMS and later VM/CMS, which stood for Control Program (Virtual Machine) / Cambridge Monitor System. The hypervisor approached used in the CP/CMS uses a minimal kernel that provides virtual interfaces to the underlying hardware. This approach only works on fully virtualizable hardware [24] or kernels that have been programmed to work around machine short comings, such as what Xen [5] has done. The rise of workstations and cheap computing saw the end of research and use of mainstream virtual computing.

In the late 1990s and earlier 2000s, virtual computing made a comeback led by VMware [4]. Virtual computing is now being used as a way to increase reliability, security, and reduce machine count. The mainstream process at this time is based upon the x86 ISA, which is not completely virtualizable due to the fact that not all sensitive instructions are a subset of privileged instructions [24]. In this case, the virtual machine is interpreted, where each instruction is converted into compatible code and executed on an emulated processor, this, however, can be extremely slow. VMware has spent considerable amount research time into making this process faster by means of dynamic recompilation [25], when code is read ahead of run-time and changed to run on the existing hardware. This enables non-virtualizable systems to be virtualized. VMwares two free virtualization environments, Player and Server, are based upon a standalone process running in a host computer, where as ESX is based upon the hypervisor concept. The main problem with ESX is that it is supported by a limited amount of hardware configurations and to this date SATA hard drives are still not supported.

There is still a considerable amount of overhead in dynamic recompilation and so research in Cambridge came up with the idea of using paravirtualization. Paravirtualization takes advantage of the fact that a large chunk of system calls does not require the usage of privileged instructions. What Xen does is change the composition of the guest operating system's system calls so that instead of executing privileged instructions they execute hypercalls that trigger the host operating system to deal with the privileged instructions. The disadvantage occurs in systems that have several system calls that trigger hypercalls over and over. Furthermore, as stated earlier, Xen does require the use of a non-standard kernel, installation of which is daunting for even experienced users. Xen is based upon the hypervisor concept, but this is for the most part invisible to the users as Xen uses drivers from Linux and is therefore as compatible with differing hardware as Linux is.

Hardware companies are recognizing the need to support virtualization at the hardware level and have begun to add instructions to assist in VM. Xen was the first software suite to show this publicly. This was followed up by a project called KVM, Kernel based VM, whose purpose was to show how efficient a VM can be thanks to hardware virtualization instructions and using already existing operating system code. Specifically, KVM use the Linux kernel's subsystems to deal with the VMM role. Thanks to this their code base is able to be kept at a minimum, while the major features in the VMM have been tested thoroughly a long time, thanks to them being a part of a well validated kernel. Recently after the introduction of hardware virtualization, VMware published a paper [26] showing that hardware virtualization was still not where their software process is.

Another aspect of virtual machines is the ability to multiplex network interface cards. This is done in two different ways, bridging and network address translation (NAT). Bridging multiplexes the device at OSI layer 2, such that the VM's network devices have an ip address on the local area network. The other, NAT, multiplexes right above the network layer, such that the virtual machine has a private address unknown to the host machines local area network. The two primary advantages of NAT are security and no loss local ip addresses.

2.3 Virtual Networking

The basis for the “Grid problem” is ”flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions, and resources” [2]. This works fine in homogeneous situations; however, the Internet does not provide a homogeneous environment given firewalls and NATs (network address translators).

Firewalls provide a layer of security by permitting and denying traffic from entering a network. NATs allow multiple hosts on a private network to share a common public address. Tightly controlled firewalls and NATs are usually made such that the host(s) are unable to receive network messages until they have made communication with a public host. This connectivity will only last so long as the firewall or NAT allows the state to exist, commonly 30 seconds. Often times these two devices are used as a means of security which TCP/IP does not inherently give, so it would be unacceptable to remove their use. The problem in this case is defined as how to gain access to resources that are behind incompatible networks. The solution is virtual private networks (VPNs) with support for TCP/IP.

The most simple form of a VPN comes in the form of central based VPNs, such as CiscoVPN [27] and OpenVPN [28]. These systems require that a user have a certificate, user name, and password to gain connectivity to the network. Several similar offerings have been provided specifically for the grid community such as ViNe [29], VNet [30], and Violin [31]; however, none of these solutions provide encrypted communication. All these technologies share a common issue that is, having a centralized system which requires administrators to setup addressing and routing rules; having this central system allows for the network to be easily compromised.

This gives motivation for distributed systems, such as IPOP (Internet Protocol over Peer to Peer - IP over P2P). IPOP is built upon the principle of peer to peer networking, where all nodes are created equally, being both server and client. In peer to peer models, new peers are started by connecting to known good peers which are listed in a preconfigured file. This initial list has no size limits and is encouraged to be very large. As long as one of those peers is alive, new connections can join the network and because the system is peer to peer, even if all the initial peers go down, already connected peers will remain connected. After connecting to the first peer, nodes attempt to discover other nodes that are close to each other.

CHAPTER 3
THE CONSTRUCTION OF THE GRID APPLIANCE

The basis for the Grid Appliance has been defined through the use of virtual machines, grid scheduling software, and virtual networking. In this chapter, the discussion focuses on implementation of these features into the Grid Appliance. Furthermore, topics including security, administration, and user interfaces are introduced and discussed in depth. No one feels safe in using products which do not present any form of system security, through techniques employed, the Grid Appliance presents multiple layers of security, which provides a truly safe environment. Without administration features, Grid Appliance systems would likely not be redeployed outside of the initial system. Providing a user interface, means even foreign users to grid environments will not feel alienated and have access to a rapid task scheduling environment.

3.1 Virtual Machines

Modern virtual machines are highly portable, encapsulated systems in a file that provide a homogeneous virtual system running on heterogeneous hosts, which can be run on any system with supporting software. As of this date, VMware has support for the three major operating systems, Windows, Linux, and MacOS (more information is available in Appendix A); and the Grid Appliance has successfully run on all three. The Grid Appliance is distributed as a compressed file weighing in at 229 Megabytes. VMs are capable of running pre-built operating systems that require no configuration from the user; because of this, the Grid Appliance is able to be used independently of the underlying hardware and software configurations. One thing not mentioned in any paper is that the homogeneous environments provided by virtual machines is limited by the instruction set of the processor, for this reason software included in the Grid Appliance is compiled to run on at least the 686 (or Pentium Pro / II) architecture. Knowing this, testing of the Grid Appliance concerns only components inside the system.

3.1.1 Virtual Machine Independence

From the ground up, the Grid Appliance was developed with a focus on being a perfect fit for generic virtual machines. This presented quite a conflict, because VMware hardware configuration slightly differs from those of rival virtualization products; further, VMware has developed a disk format that has only recently been opened to the public, but not much work on compatibility has been done[1] . To date, the Grid Appliance has successfully booted in VMware and Xen and testing will soon begin for Qemu, KVM, Parallels, and VirtualBox.

3.1.2 Data Portability

As mentioned earlier VMs provide encapsulation in the form of VM disks, the Grid Appliance takes advantage of that by supporting multiple different layers of disk access provided by UnionFS [32]. The idea being that there are three layers, base operating system layer, developers layer, and a users layer. These layers or stacks are all combined into a single file system from the users perspective, see figure 3-1. The base layer contains the core configuration of Grid Appliance and can be shared by multiple virtual machines reducing disk space costs, this layer is read-only. The developers or site specific layer is used by local system administrators to add special functionality to the Grid Appliance that is not provided in the default image. To access this a user need only select at boot, that they would like to go into development mode. At the end of the session, the developer runs a script, which blanks out any Grid Appliance configuration and is able to release this new image to users. The user layer affords the ability for users to migrate their data from one machine to another without worrying about the underlying configuration of that Grid Appliance.

[1] A utility provided by the Qemu group called qemu-img allows for the conversion of single file, growable VMware images; however, this is only one of the possible 4 different VMware disk configurations.

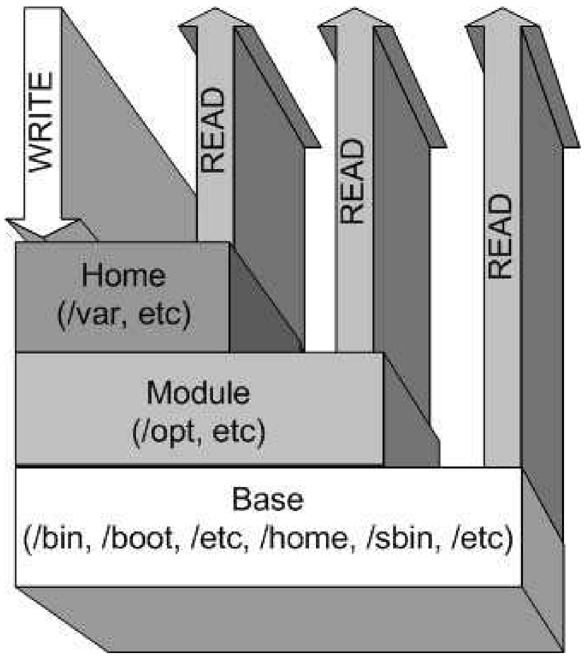


Figure 3-1. UnionFS layout in the Grid Appliance.

3.2 Condor

The decision to use Condor is based on it being open source, extremely fault tolerant, and easily configurable. Since previous sections have gone into detail about Condor features, this section focuses on the configuration in the Grid Appliance. One issue with Condor is that it binds to an IP address instead of a device. Because of this inconvenience, a script in the Grid Appliance occasionally checks the IP address and restarts Condor if needed with the new address. To make a Grid Appliance a manager, worker, or submission only node requires only the creation of a specific file. As the Condor scripts created for the Grid Appliance start, they will check this file and start Condor appropriately. A similar feature is used for the condor manager's IP address, which is stored in a file so that users can easily change that configuration value as well.

To validate the Condor installation the following tasks are performed:

- batch execution of 100 standard output jobs,
- test case of Condor's DAG scheduler,
- check pointing jobs,

- Condor flocking [2] .

3.3 IPOP

Because the Grid Appliance uses IPOP, there is very limited centralized management of the virtual network. A detailed discussion of the IPOP architecture is addressed by related work [33, 34], this paper will lightly touch IPOP architecture prior to the Grid Appliance and changes made for the Grid Appliance and user-friendliness. For a reliable and self managing network three services were added to IPOP they include DHCP (Dynamic Host Configuration Protocol), DNS (Domain Name System), and ARP (Address Resolution Protocol).

3.3.1 Architecture

IPOP contains four major assemblies: Brunet, libtuntap, IPRouter, and SimpleNode; this is done to separate major features from each other. The Brunet library contains the peer to peer connectivity. Libtuntap is used to read and write from the virtual network adapter, TAP [35]. IPRouter binds libtuntap and Brunet such that it is responsible for sending and receiving data over Brunet and interfacing with libtuntap. SimpleNodes are bound to Brunet and are used to setup static peers for inclusion on the initial list of IPOP peers from which other nodes connect to the system. The traditional mechanism for deploying IPOP can be seen in figure 3-2. The application sends and receives on the TAP device, while IPOP reads and writes to the TAP device. Incoming packets are received by IPOP and written to TAP. Outgoing packets are read by IPOP, converted, and sent over the physical Ethernet device. The IPOP virtual network address space is 10.128.0.0/255.128.0.0 in this example.

One of the major problems of peer to peer technologies is that NATs and firewalls tend to make it difficult. To get around this, Brunet employs UDP (user datagram

[2] Condor flocking allows a job submitter to schedule jobs to execute on another Condor managers system, when all the nodes on the submitter's system are occupied.

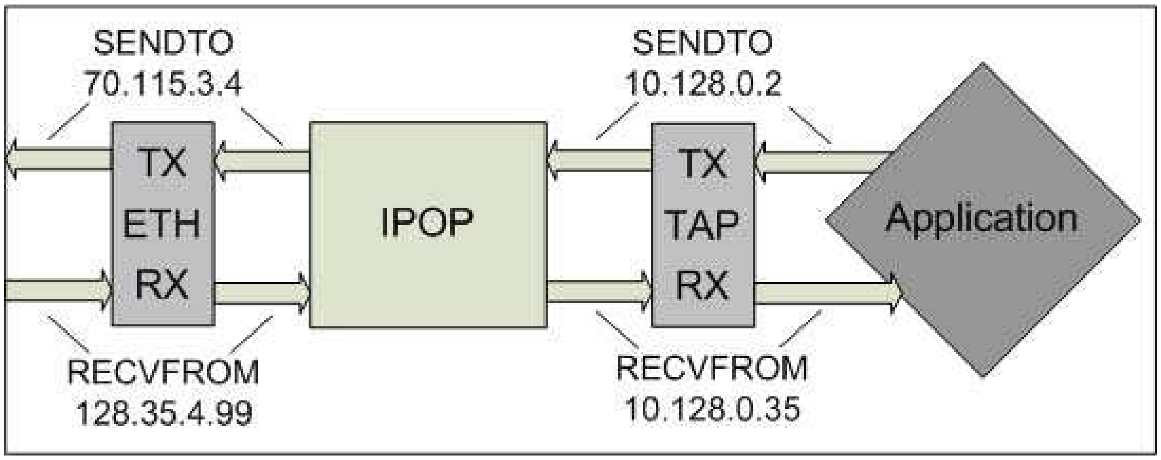


Figure 3-2. IPOP deployed in the same domain as the VM.

protocol) and NAT hole punching technologies. Furthermore, if nodes go down, Brunet is able to reconnect the system without user interference, making it self-healing. Now focus switches to the architecture that is relevant to the Grid Appliances internals, the local network stack and remote network stack negating what happens in the peer to peer networking overlay, ie Brunet.

In the original papers discussing IPRouter [33, 34], it required a lot of user input to successfully start. This is because it needs to know details such as the hardware and IP addresses of the TAP device, that is, there was no true DHCP process. This also meant that the TAP device needed to be completely setup prior to starting IPRouter. A lot of focus went into making it such that the only requirement for starting IPRouter was that a TAP device existed on the system. Now IPRouter learns these addresses by reading them out of the packets going over the TAP device.

By enabling a robust IPRouter that can learn its addresses allows for dynamic changes to system configurations, such as a change in IP address; furthermore, this allows for two interesting possibilities: IPRouter running in a different execution domain (Figure 3-3) than the virtual network communication and allowing multiple virtual network adapters per IPRouter. The application sends and receives over its logical Ethernet device, which is revealed to the host as a VIF (virtual interface). The IPOP bridge connects the

VIF to the TAP device, which IPOP reads and writes. IPOP sends and receives packets for the 10.128.0.0/255.128.0.0 virtual network over the host's physical Ethernet device. At this point in time, only the first feature has been implemented and checked. The second feature is similar to how the ViNe router [29] allows for multiple machines behind a single domain communicate over the regular local area network adapter.

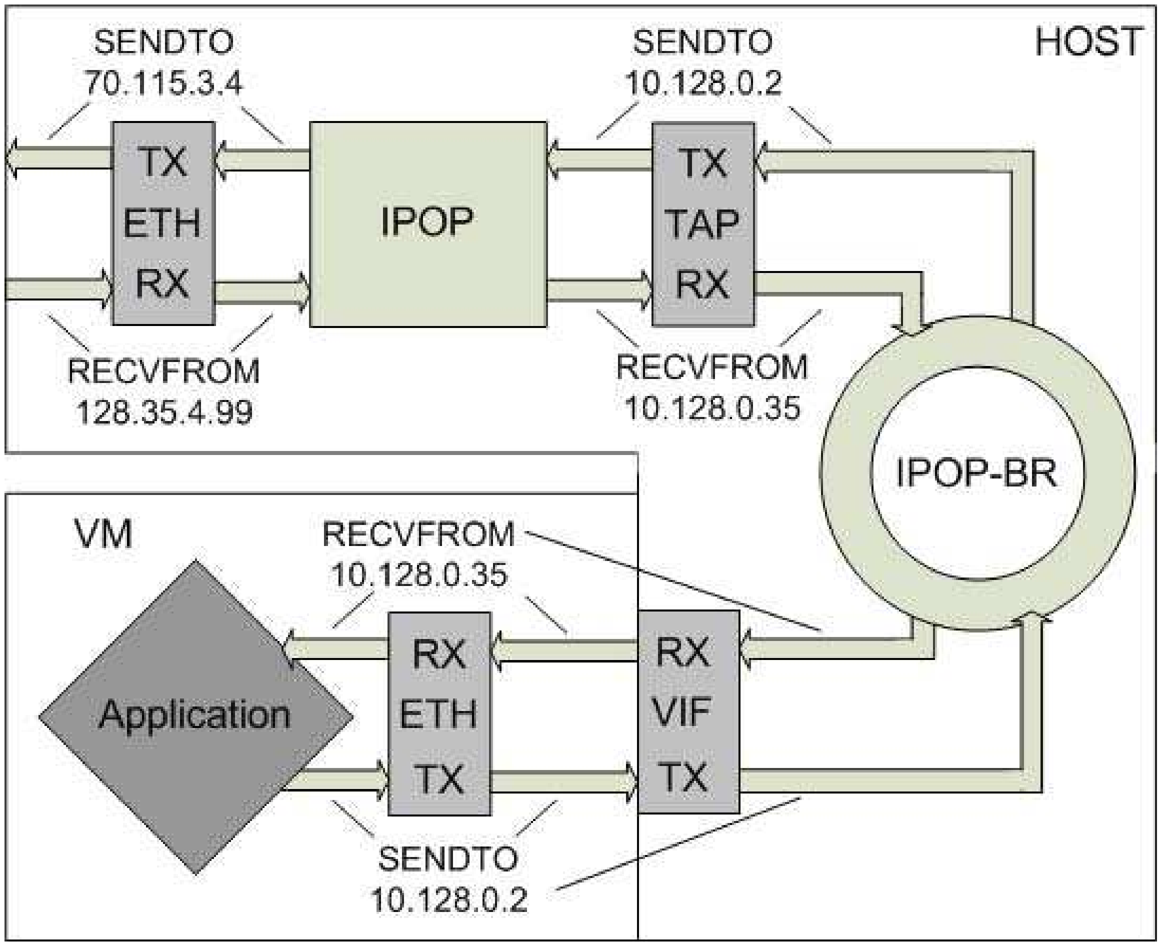


Figure 3-3. IPOP deployed in a separate execution domain from the VM.

The work focused on running IPRouter in a different execution domain includes testing using Xen and VMware. In Xen, the idea is to replace Xen's bridging rules to use the TAP device instead of the Internet based network devices, typically eth0. In VMware, this is done by bridging the host only network connection to the tap device. In these cases, the operating system used was Linux and the tap device is used primarily to bridge data into IPRouter and the actual virtual network interface in the virtual machines connects over the virtual network (IPOP) without the VM's knowledge.

The final portion of IPOP is the libtuntap, which contains operating system dependent code for communication with the TAP device. The interface provided has been written in such a way that the same IPRouter and Brunet assemblies can be run on any system as long as there is a libtuntap library available for these systems. To date, this has been coded for Linux and Windows.

3.3.2 Services

In creating a self-configuring grid environment two important services are required, those being DHCP [36, 37] and DNS [38, 39]. Both of these are found in IPOP. DHCP or dynamic host configuration protocol allows for the assigning of a unique IP address from a central server. Domain name server provides a way to map names of machines to their IP addresses and vice-versa. In the original IPRouter, the local machine needed routing rules to access the network, to get around this, IPRouter now features an ARP [40] service.

In IPOP, there are two forms of DHCP, one that works similar to the standard format using a centralized server with communication using SOAP (Simple Object Access Protocol), while the other works in a distributed method using distributed hash table [41] creates to obtain IP addresses. The former has been in place since Summer 2006 and will eventually be phased out as the latter becomes more stable. When IPRouter receives a packet from the TAP device that has DHCP port numbers associated with it, it realizes this is a DHCP packet. This is allowable because the DHCP port numbers have been defined in a RFC [37] and reserved by the system for these purposes specifically. Libraries have been written to support the decoding (and later encoding) of these types of packets. Depending on if SOAP or DHT (distirbuted hash table) DHCP is enabled, will result in different affects. In the case of SOAP, the client will send the SOAP server a request and the SOAP server will map an IP address to the requesting client with a reconfigurable lease time. In the case of DHT, the client attempts to use Brunet's DHT [41] feature to

IP Address	Host name
10.128.0.1	C128000001

Table 3-1. IP Address to hostname mapping using IPOP's dns server.

make exclusive creates in the DHT space to reserve an IP address, if this is successful, the node has obtained that IP address. The major benefit of DHT is that it is distributed and does not contain a single point of failure; however, DHT requires that the Brunet system be in a very good condition, where all nodes are accessible to all nodes. Thus the SOAP option is still viable, until there is enough stability in the Brunet system. By this mechanism, the DHCP clients that exist in current operating systems are capable of setting the IP address of the TAP device, further reducing user configuration.

Many classic grid applications require the use of DNS hostnames, thus motivating the creation of the DNS service. This service is executed outside of the IPOP execution domain for modularity purposes and is written in Python. One problem noticed when building the DNS and DHCP system was that by default the DNS server's list is overwritten by the most recent DHCP call. For Linux, this has been fixed by the use of the package resolvconf; however, later version of resolvconf have made it such that if a DNS server is running on the localhost, no other DNS servers are needed. This problem required the use of an older version of resolvconf than provided in the Grid Appliance current operating system. In older incarnations of the Grid Appliance, an edited hosts file was used instead of DNS, since there must be a mapping for each host name to IP address, this file could be in excess of megabytes, using the DNS server method name lookups were shrunk from measurable seconds to being milliseconds. The DNS server used is of a simple design, where the IP address is mapped directly to a single IP for an example see table 3-1.

Prior to the inclusion of ARP the user would need to add a rule to the routing table to add a gateway, also, this machine needed a fake hardware address added to the ARP cache. The new implementation responds to ARP requests, so that the nodes now think they are on the same layer 2 network.

3.4 Security

In providing a public grid system, security is of the utmost importance. The Grid Appliance's goals in this arena are to prevent users of the Grid Appliance from affect the host machine and from affecting remote machines. The two areas focused on in the Grid Appliance are sandboxes and centralized, tightly controlled systems. Also it is important to provide the ability to create completely private Grid Appliance systems, which focuses on the use of encrypted channels.

Sandboxes [42] provide truly isolated computing environments at the cost of limited usability. The fundamental idea is that once the task is crunching to not let anything in or out until the task is completed. This makes true sandboxes non-ideal for user interaction and positions them as execution environments only. This is done by removing any network interfaces from the virtual machine.

Having a centralized environment makes management simple, because the administrator can easily see when users are misbehaving and deal with it accordingly. The downside presents itself when the system's servers are hacked either taking user data or more likey a denial of access which essentially causes the servers to go offline. If this were the case, the execution environments would complete jobs and have no where to send them, possibly causing the results to be lost; furthermore, no new jobs could be added, since users would be unable to connect. Also user data would be unavailable during this down time.

Taking into idea of the sandbox, the Grid Appliance employs virtual machines, firewalls, virtual networking, and IPsec to create a sandbox. VMs provide an abstracted unit from the underlying system. Firewalls along with the virtual network ensure that transmission of incoming and outgoing Internet traffic pertains specifically to the grid system. IPSec makes it such that only trusted nodes are allowed into secure pools. This approach is significantly different from that proposed by OurGrid [43], which uses a complex system for deploying tasks to VMs and receiving files over NFS.

3.4.1 Virtual Machines

If a VM is ever overtaken the worst thing an intruder can do is compromise the data inside the virtual machine and cause the virtual machine to behavior erratically, this is because of the abstraction between VM and host machine. In computers there are at least two levels of privilege: system and user modes. The system level has control over everything and can only be affected by system privileged tasks; where as the user mode can be controlled by all levels, however, the operating system kernel usually create independent execution platforms for each application. VMs execute in the user mode.

There is a lot of difficulty in keeping applications from effecting each other, thus brings the desire to have applications running in different domains altogether, this can be achieved using a VM. The VM is software that runs in user mode but allows for guest operating systems to run on a virtual cpu believing that they are the sole owner the underlying processor. This makes it such that processes inside the VM do not know about processes outside running on the host and vice-versa. Therefore software bugs in the VM can not affect software running outside the VM. Of course this puts some reliance into the VMM being a secure and stable software. VMMs have an advantage over operating systems lies in their simplicity. Also most VMM software developers are constantly looking for holes and fixing them as soon as possible.

The VM still allows for users to take advantage of resources given to them by the VMM, such as Internet connected network devices and the ability to run dangerous processes. Network issues are covered in the next two sections and the effect of dangerous processes is ameliorated by the existence of tweakable parameters for VMMs that allow users to specify memory and processor allocations to specific VMs. A user can also shutdown a faulty VM at any time.

3.4.2 Firewalls and Virtual Networking

In order to prevent communication from outside the Grid Appliance to inside and vice-versa, a strict firewall is implemented. A firewall is a super user level application that deals as a middle man in network communication. In the simplest form, it takes in either a send or a receive checks to see if the transmission should be accepted based upon one or more of the following parameters: sender port, receiver port, sender ip address, receiver ip address, local transmission device, and protocol.

Firewalls by themselves are limited because there still needs to be open ports and protocols allowed for grid software to work. The approach taken by the Grid Appliance is to consolidate all grid software traffic to run over a virtual network, IPOP, which runs over a single network port and protocol and blocking via the firewall all other network traffic. Since the operating system will prevent new processes from using a bound port and all other ports are closed, only a super user could start new network services. Thus security falls into how well defended is the super user access.

In the Grid Appliance there are only two ways to obtain super user access, know the password of the local user or have the administrator's ssh certificate and password; furthermore, the only way to access root remotely is via ssh server that requires the use of certificates to obtain access. The purpose for the administrator's ssh is discussed in more detail in Section 3.5. and will probably be removed once the project is more mature.

Another form of virtual networking used is the host only network interface provided by VMware, which allows the creation of network adapters in guest operating systems that can only communicate with the local host. This configuration allows all forms of networking to transmit across it. This is further discussed in Section 3.6.

3.4.3 IPsec

The basis for IPsec (IP security) [44] is that communication amongst a group of nodes occurs in an encrypted and secure manner. IPsec is based on the SSL (secure sockets layer) method. There is a two step process for incoming nodes into an IPsec system, obtaining a signed certificate and hand shaking with other nodes in the pool.

Consider a system whose ”Certificate Authority's” name is Alice to which a new node, Bob, would like to connect. Bob starts by creating a certificate request and includes in it a unique identifier, perhaps his land line number. This would be sent to Alice, who will make a decision on whether or not to sign the request and return to Bob. If Alice approves, she will send Bob back a signed certificate and her own public certificate that will gain him access to the system.

With this certificate, Bob can now talk to other members of Alice's system. Maybe he wants to talk to Carol, because Carol has some unused resources that Bob would like to take advantage of. Bob begins by contacting Carol by stating he wants to begin secure communication. The first part of the operation is to coordinate that the users are who they say they are through the use of their signed certificates and Alice's public certificate. The idea being that if Carol asks Bob for his phone number and compares that to what the caller's identification states and follows that up with the reasoning, the caller ID says its Bob, Alice trusts that it is Bob, so therefore I trust Bob. The second phase involves Carol and Bob setting up encryption and decryption keys over which they will pass their data. This communication ends when either Carol or Bob decides to end it or they go for a long period of time without talking and they are disconnected.

What the method above lacks is the description of an automatic method for Bob to get Alice his certificate request and Alice in turn to respond to Bob's request. In the Grid Appliance, the method is still being refined but works by starting Bob in an insecure pool. At any time, Bob can start a script that will automatically create the certificate and send it to Alice. Alice will be presented with Bob's credentials and will simply click an accept or deny button which if accepted will automatically sign Bob's certificate and send it and Alice's public certificate back. At which point, Bob will be transfered into the secure pool where only certified nodes can play. While it is entirely possible to allow Alice to automatically accept nodes, at this point in time, the option is still left as a conscious decision for Alice to make.

To summarize, IPsec provides a secure way for a system of nodes to talk amongst each other using a common public certificate, given by Alice. Nodes can not impersonate other nodes unless they obtain the signed certificate from Alice and obtain the nodes address. Furthermore, nodes can be excluded from pools via an interface to Condor or by adding them to the exclusion list included with Alice's certificate. In Grid Appliance the phone number is replaced by the virtual IP address allowing point-to-point authentication in a simple scalable manner. This technique is ideal for creating independent, private pools of Grid Appliances also for ensuring user authentication in public systems.

3.4.4 The Fall Back

In the case that all else fails and the network becomes uncontrollable there are many methods by which the system can be retaken. This all depends on the level of corruption, but as a fail safe scenario, it is possible to restart the entire grid system within minutes thanks to the convenient packaging via the system in a file configurations presented by virtual machines, this of course only works for nodes, which are under direct control of the administrator. The downside is that nodes not directly in control of the main system administrator would need to restarted by their local administrator. However, there is a significant amount of easy to use scripts that will stop and delete bad instances and create and start clean instances. As another step, the Brunet namespace could be changed such that nodes that were infested would be unable to reconnect to the now clean system. This is a scenario that will most likely never be seen but has been prepared for in such an emergency.

3.5 Administration

Management of clusters is not trivial and adding the complexity of a distributed system complicates the matter more. In most clusters, different machines are assigned some unique identification by the administrator so that if the machine acts up the administrator can easily identify the machine acting up and diagnose the problem. In the case of Grid Appliance, this is impossible, since nodes are given IP addresses via dhcp as well as requiring the deployment of SimpleNodes for initial. The problem has been further extended by allowing users to create add new nodes to the system at will.

To deal with these issues, the Grid Appliance has been designed to be self-healing. In cases, where a machine is removed from internet access for long periods of time, IPOP will continue to seek connectivity to nodes until Internet access is restored. If jobs are lost over Condor, then they will be re-executed upon connecting back to the server. In fact, the jobs are given a 2 hour window for nodes to reconnect before they are dropped from the node executing them. Updates are handled through the Debian's "dpkg" interface and will be rolled back upon failure. As in conventional physical machine environments, there are situations in which the system state is such that it requires a reboot to re-initialize the virtual machine; however, if the disk is corrupted, there is no other solution besides starting a new virtual machine. That is as simple as copying new files over the old virtual machines files.

There has also been investigations into how to provide support for a global administrator. The primary method thus far relies on each Grid Appliance having a ssh client that runs only on the virtual network and accepts only a pre-defined ssh key. An administrator can access any virtual machine in the pool and diagnose issues as needed. This approach has also been used in Planet Lab. The main downside to this is that if the problem lies in IPOP then this approach is useless. As a work around, it has been proposed that there be an external application that runs on the host, which could provide an ssh tunnel directly into the virtual machine. There are also scripts available which help in the deployment of the Grid Appliance on systems that run VMware Server and have ssh connectivity.

Finally, Condor provides a way to check the status of the current pool by running "condor_status" application. An administrator can monitor the count of nodes to ensure that machines are able to connect, stay connected, and are functioning properly. Other management features of Condor include "Condor Quill" that provides a history of job submissions and "CondorView" that provides a graphical representation of current and past utilization.

The work done so far is only an initial foray into the topic of administration and since the environment of the Grid Appliance is still in development, this topic has not been covered in as much depth as possible.

3.6 User Interfaces

Traditionally access to grids has been limited to tools like SSH, telnet, SFTP, and FTP. In the worst case, these tools require experience with a console and know how of the special commands for these tools, while SFTP and FTP have the benefit of having GUIs coded for them to make it easier to navigate file systems. Where as SSH and telnet force users to remain in the console and learn the hosted operating systems command line interface. All these systems also require that a user be given direct access to remove machines by adding user account on the local machine or network. This chapters focus is discussion of user interfaces as an enabler for regular users on grid systems.

3.6.1 Application Access

To access most grid resources a user most be comfortable with a console environment as the majority of tools are only available there. This can provide a large and undesirable learning curve for users that are only interested in a single application that runs with different data sets. By having a user interface, the grid environment becomes more accessible and hence should have a diverse population of users. Two examples of grid systems providing user interfaces are InVIGO [45] and NanoHub [46].

Most grid systems offer only user control features and system overview in a user-friendly interface. This is the case for sites like TeraGrid Portal [47] and PlanetLab. Even Condor provides utilities to easily view system statistics, as described previously "CondorView" [48]. The challenge faced by grid systems is that there is no single user interface for the vast library of user applications. The issue is complicated by the difficulties in user interface programming.

InVIGO providess a user front-end for job submission; however, developers for interfaces are forced into working with form based, static content and a limited API. The concept is good; however, InVIGO is difficult to port given that most features have been designed from the ground up to support InVIGO. Users prefer more lively and dynamic content, which at this point in time can only be provided either through a Java or Flash runtime or natively through Javascript, none of which are supported in InVIGO.

NanoHub relies on VNC [49] sessions using Rappture [50]. VNC provides a desktop session on a remote computer, where tasks execute remotely but results are displayed locally. Rappture is a Tcl/Tk based application which takes the input of an XML (extensible markup language) file and displays a graphical user interface. The user is instructed to give some inputs and submit for execution. This calls up a script, which is specified in the XML file to execute given the variables and return the result. These scripts can be written many of the most popular languages such as Perl, Python, C, and Tcl and is easily portable to other languages. The two main advantages of Rappture is the simplicity of disambiguating displayed content from the task execution and providing a easy to work with framework that does not require the user to code and graphical user interface content.

OurGrid software suite includes MyGrid, which provides a user interface for submitting jobs over the Internet. The GUI portion of it provides tabs for adding new jobs, which come in predefined files; status of current jobs, and status of the system; however, according to the latest documentation does not run on Windows.

By itself, the Grid Appliance provides an X11 based windows manager from which the user interacts with their system. Furthermore, the web interface for the Grid Appliance also supports VNC sessions. The main benefit of this is that users can code for the native APIs, whether they be Windows classes, Windows Forms, X11, TK, GTK, QT, etc, rather than web coding. Graphics APIs by nature provide dynamic presentations and do not require the developer of an application to adapt already written software for a web interface. Furthermore, the Grid Appliance employs Rappture and provides a library that make it easy to submit Rappture jobs to a scheduler. One other good feature of VNC is that by default the sessions will not expire until they shut are off.

The web interface for the Grid Appliance also provides for AJAX [51] based user interfaces. AJAX or asynchronous javascript over XML provides a way to create interactive web applications that are supported in most web browsers since the early 2000s, including Internet Explorer, Firefox, Mozilla, and Opera, to name a few. The benefit of this over VNC is that the javascript runs on the client side reducing the extra resources required for VNC sessions, which have been measured to be at least 10 MB of main memory per instance. Furthermore, when the server machine crashes there is no way to save the state of a VNC session; however, the user variables in an web page are easily stored to disk and provides for session handling even in the case of a hardware failure.

Local users are given multiple ways to interface with the machine, as mentioned before, there is a default X11 based windows manager called IceWM, which is similar to Windows 9x user interface. Users can also access the system through SSH (secure shell) for those desiring not to be confined to a graphical user interface. Work is underway to make each Grid Appliance capable of hosting web interfaces only to the local host, this is going to be based off the web interface in a module based format.

3.6.2 Data Access

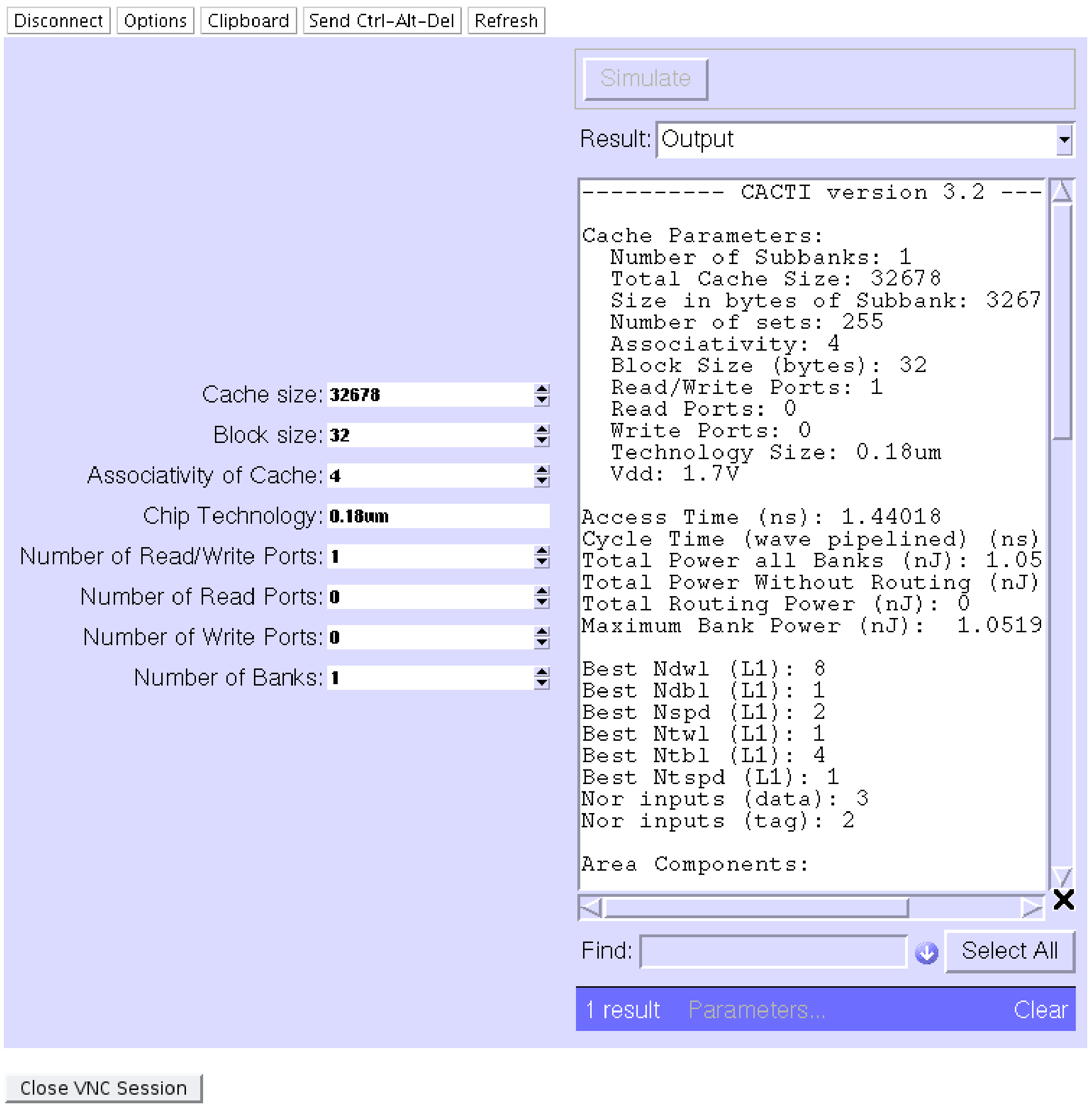
Another major facet of user interfaces is access to user data, InVIGO provides a web interface to the users file system, where NanoHUB uses DavFS[]. Another widely accepted remote file sharing platform is Samba, which is the default file sharing system for Windows.

The only way to access user files through InVIGO is by accessing it through the web, through a perl application called Drall [52]. The downside to this is that the user is forced to download any files prior to accessing them locally. Furthermore, it is very difficult to write scripts that would automatically retrieve files. The positive side to it; however, is that InVIGO developers do not need to concern themselves with how to ”mount” their file system in other operating systems, because as long as their is a web browser, their file system works.

DavFS [53] is a HTTP (hypertext transfer protocol) file system. This allows web systems to integrate DavFS directly into web interface and take advantage of the user data there as opposed to requiring extra user accounts through software such as PAM (pluggable authentication modules) and LDAP (lightweight directory access protocol), similar to what FTP (file transfer protocol) and SSH require. Currently to access DavFS shares in Windows, a user needs to adding a new remote folder, which is not trivial for regular users, and in most flavors of Linux, which require the remote folder be mounted, which is even more complicated than the Windows version. On the positive side though, there are many web applications that can support reading a DavFS from the Internet, remote DavFS clients.

The most widely integrated file system to date is Samba, which is the basis for Windows file sharing. The downside of Samba is that the versions supported on Windows have a weak level of security and they too require integration with a password authenication system like FTP and SSH. Samba's saving grace is that a client is integrated into almost every modern file system manager to date

Given this wealth of options, the Grid Appliance supports these all these various data access modes. File sharing for the web interface uses DavFS and includes a web interface to access files, these are based on implementations for PHP. The advantage of DavFS in this system is that virtual user directories were setup for individuals as they register accounts on the web interface; the DavFS server used in the Grid Appliance is made specifically for this purpose. This method provides a secure file system without requiring direct access to the remote machine. For local instantiations of the Grid Appliance, users are able to access their files through Samba without knowledge of a password or other complications.

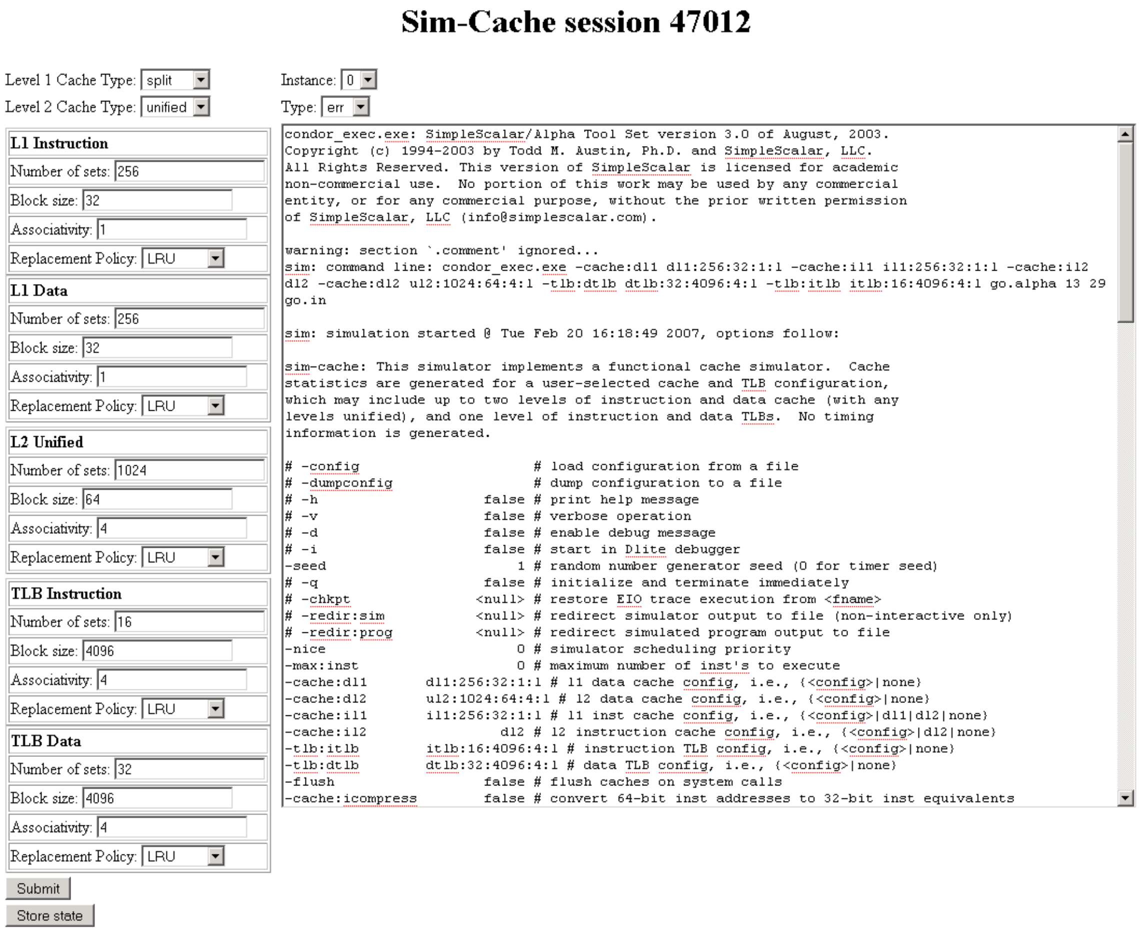


Result: Output

Cache size: 32678
Block size: 32
Associativity of Cache: 4
Chip Technology: 0.18um
Number of Read/Write Ports: 1
Number of Read Ports: 0
Number of Write Ports: 0
Number of Banks: 1

```
---------- CACTI version 3.2 ---

Cache Parameters:
  Number of Subbanks: 1
  Total Cache Size: 32678
  Size in bytes of Subbank: 3267
  Number of sets: 255
  Associativity: 4
  Block Size (bytes): 32
  Read/Write Ports: 1
  Read Ports: 0
  Write Ports: 0
  Technology Size: 0.18um
  Vdd: 1.7V

Access Time (ns): 1.44018
Cycle Time (wave pipelined) (ns)
Total Power all Banks (nJ): 1.05
Total Power Without Routing (nJ)
Total Routing Power (nJ): 0
Maximum Bank Power (nJ):  1.0519

Best Ndwl (L1): 8
Best Ndbl (L1): 1
Best Nspd (L1): 2
Best Ntwl (L1): 1
Best Ntbl (L1): 4
Best Ntspd (L1): 1
Nor inputs (data): 3
Nor inputs (tag): 2

Area Components:
```

Figure 3-4. VNC session powered by the Grid Appliance web interface running CACTI.



Sim-Cache session 47012

Level 1 Cache Type: split
Level 2 Cache Type: unified

L1 Instruction

Number of sets:	256
Block size:	32
Associativity:	1
Replacement Policy:	LRU

L1 Data

Number of sets:	256
Block size:	32
Associativity:	1
Replacement Policy:	LRU

L2 Unified

Number of sets:	1024
Block size:	64
Associativity:	4
Replacement Policy:	LRU

TLB Instruction

Number of sets:	16
Block size:	4096
Associativity:	4
Replacement Policy:	LRU

TLB Data

Number of sets:	32
Block size:	4096
Associativity:	4
Replacement Policy:	LRU

Instance: 0
Type: err

```
condor_exec.exe: SimpleScalar/Alpha Tool Set version 3.0 of August, 2003.
Copyright (c) 1994-2003 by Todd M. Austin, Ph.D. and SimpleScalar, LLC.
All Rights Reserved. This version of SimpleScalar is licensed for academic
non-commercial use.  No portion of this work may be used by any commercial
entity, or for any commercial purpose, without the prior written permission
of SimpleScalar, LLC (info@simplescalar.com).

warning: section `.comment' ignored...
sim: command line: condor_exec.exe -cache:dl1 dl1:256:32:1:l -cache:il1 il1:256:32:1:l -cache:il2
dl2 -cache:dl2 ul2:1024:64:4:l -tlb:dtlb dtlb:32:4096:4:l -tlb:itlb itlb:16:4096:4:l go.alpha 13 29
go.in

sim: simulation started @ Tue Feb 20 16:18:49 2007, options follow:

sim-cache: This simulator implements a functional cache simulator.  Cache
statistics are generated for a user-selected cache and TLB configuration,
which may include up to two levels of instruction and data cache (with any
levels unified), and one level of instruction and data TLBs.  No timing
information is generated.

# -config                     # load configuration from a file
# -dumpconfig                 # dump configuration to a file
# -h                    false # print help message
# -v                    false # verbose operation
# -d                    false # enable debug message
# -i                    false # start in Dlite debugger
-seed                       1 # random number generator seed (0 for timer seed)
# -q                    false # initialize and terminate immediately
# -chkpt               <null> # restore EIO trace execution from <fname>
# -redir:sim           <null> # redirect simulator output to file (non-interactive only)
# -redir:prog          <null> # redirect simulated program output to file
-nice                       0 # simulator scheduling priority
-max:inst                   0 # maximum number of inst's to execute
-cache:dl1       dl1:256:32:1:l # l1 data cache config, i.e., {<config>|none}
-cache:dl2       ul2:1024:64:4:l # l2 data cache config, i.e., {<config>|none}
-cache:il1       il1:256:32:1:l # l1 inst cache config, i.e., {<config>|dl1|dl2|none}
-cache:il2                  dl2 # l2 instruction cache config, i.e., {<config>|dl2|none}
-tlb:itlb        itlb:16:4096:4:l # instruction TLB config, i.e., {<config>|none}
-tlb:dtlb        dtlb:32:4096:4:l # data TLB config, i.e., {<config>|none}
-flush                  false # flush caches on system calls
-cache:icompress        false # convert 64-bit inst addresses to 32-bit inst equivalents
```

Figure 3-5. AJAX session powered by the Grid Appliance web interface running SimpleScalar.

CHAPTER 4
RELATED WORK

The Grid Appliance is not the first project that has attempted to put a grid system into a virtual machine as a redeployable image. Similar projects include Globus Workspaces and OurGrid. Other projects have also sought to bring grid computing in a non-intrusive way notably the Minimal intrusion Grid.

Globus Workspaces [54, 55] are an abstract unit that are not a one size fits all solution, proposing customized VMs for different applications. There is some discussion on connectivity regarding certificates but not on how they are distributed nor how well VMs must be connected to interoperate. Their grid system is based upon Globus Toolkit 4 [23]; how this is configured is not discussed. On a final note, the purpose of the Globus Workspaces is to run in a well defined environment as software must exist to start, stop, and pause the VMs as there is no discussion of direct user interaction. Thus a logical conclusion that adding new execution machines is not a simple, distributed process.

OurGrid [43]proposes similar goals to that of the Grid Appliance, a peer to peer oriented grid structure that makes it easy to use grid resources. The OurGrid project has three different tiers, middleware managers called OurGrid Peers, user interfaces called MyGrid, and execution nodes called SWAN (sandbox without a name). The OurGrid Peers help enable the peer to peer system and broker fair trade amongst the different sites. MyGrid and SWAN were discussed earlier in the chapters regarding user interface and security, respectively. The Grid Appliance system requires only two entities, that being the SimpleNodes which are maintained by the main administration units of the system and the Grid Appliance itself. Users can turn any machine into an execution node without reconfiguring the system as is required for SWAN. Grid Appliance the local installation capabilties of the MyGrid; however, there is a firm belief that any nodes submitting jobs should also be executing them as well.

Minimal intrustion Grid (MiG) [56] proposes the creation of new grid software leaving behind existing technologies. The advantages with this approach are a simplified adaptable code base providing the minimal feature set that the grid system requires. This is one of the downsides of the Grid Appliance, containing so many different projects means that the size and complexity of the system are increased; however, the benefit of using standard software sets is their maturity and features that would be difficult to add to new software. The MiG project has recently released execution nodes that can run as Windows screen saver, akin to the SETI [57] and FOLDING [58] at home projects. The basis for trust is the use of SSL certificates, which can only be obtained directly from their website. There is no discussion on creating independent MiGs, nor, the safety of execution nodes from hostile software; however, the website references the screen saver as a sandbox. Submission occurs from the website, an idea similar to how the Grid Appliance Web Interface works. There is no discussion of how jobs are scheduled or how scheduling occurs. They make claims of decentralized grid systems but do not go into details on the matter. The project seems to have stagnated since April 2006.

CHAPTER 5
SYSTEM VALIDATION AND PERFORMANCE

In this chapter, an overviews the validation and performance evaluation of the Grid Appliance system.

5.1 Validation

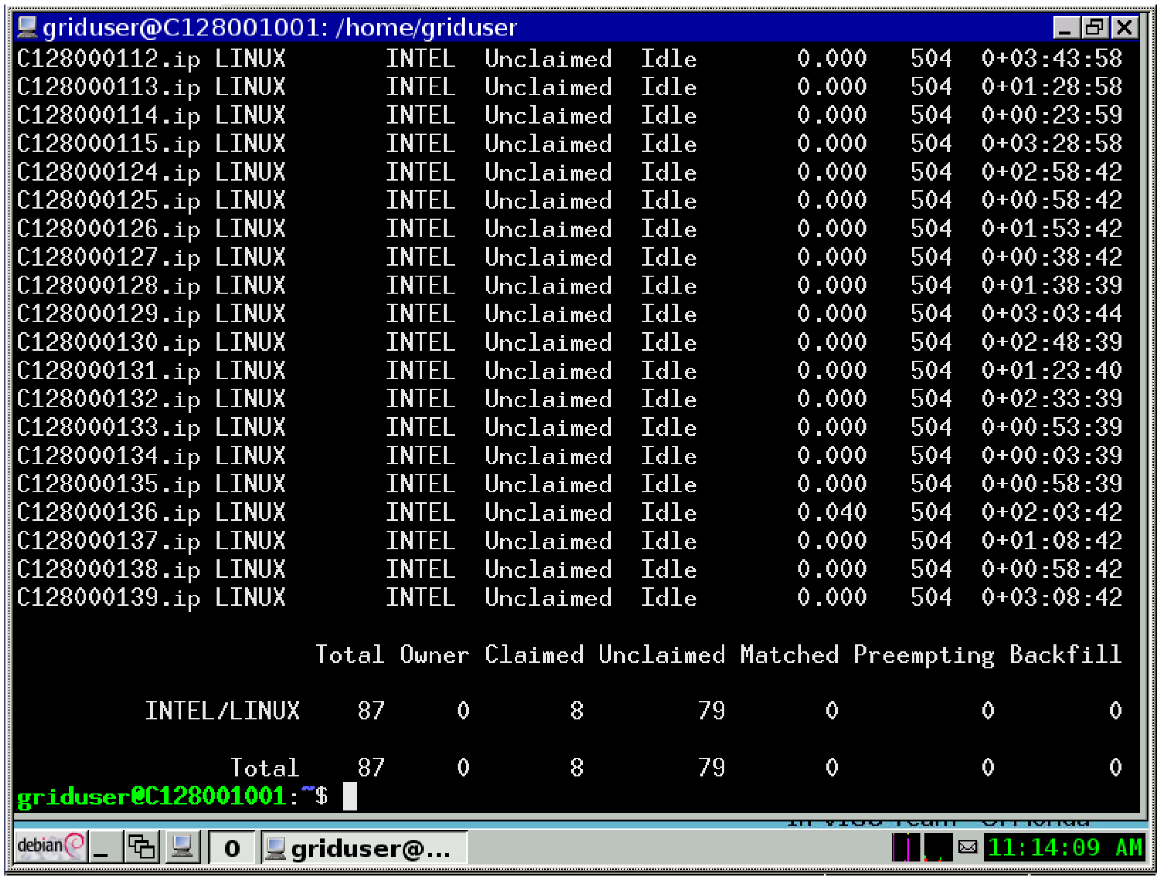
Having a large distributed system can be a nightmare to ensure that there is good connectivity from all nodes in the pool. In fact, there can be users which start an instance that may go unreported because it is unable to connect. To guarantee at least some sanity in the system tools and methods have been developed which test the connectivity of all workers to the master, all nodes to each other, and the state of SimpleNodes running on Planet-Lab. Just because the nodes have good connectivity does not mean that Condor is working, to ensure that it is, the tasks outline in 3.2 are performed. The focal point of this section is on the state of the distributed system.

5.1.1 Condor

Through the use of "condor_status", the state of all currently connected nodes are listed (Figure 5-1). The idea is to watch the change in system size to determine if there may possibly connectivity issues. Using the Grid Appliance Web Interface, this information is already available from a web site, making the task of watching the status of the system very easy.

5.1.2 Ping Test

Just because a system is responding well to the manager node, which is what "condor_status" tells us, does not mean that the system is well connected. For that case, a script has been deployed on all Grid Appliances that pings every node in the system 3 times every twelve hours to determine the connectivity. Thanks to the work of Professor P. Oscar Boykin, the task of obtaining relevant data out of the ping test's logs has been made much easier (Table 5-1).



```
griduser@C128001001: /home/griduser
C128000112.ip LINUX        INTEL  Unclaimed  Idle       0.000   504  0+03:43:58
C128000113.ip LINUX        INTEL  Unclaimed  Idle       0.000   504  0+01:28:58
C128000114.ip LINUX        INTEL  Unclaimed  Idle       0.000   504  0+00:23:59
C128000115.ip LINUX        INTEL  Unclaimed  Idle       0.000   504  0+03:28:58
C128000124.ip LINUX        INTEL  Unclaimed  Idle       0.000   504  0+02:58:42
C128000125.ip LINUX        INTEL  Unclaimed  Idle       0.000   504  0+00:58:42
C128000126.ip LINUX        INTEL  Unclaimed  Idle       0.000   504  0+01:53:42
C128000127.ip LINUX        INTEL  Unclaimed  Idle       0.000   504  0+00:38:42
C128000128.ip LINUX        INTEL  Unclaimed  Idle       0.000   504  0+01:38:39
C128000129.ip LINUX        INTEL  Unclaimed  Idle       0.000   504  0+03:03:44
C128000130.ip LINUX        INTEL  Unclaimed  Idle       0.000   504  0+02:48:39
C128000131.ip LINUX        INTEL  Unclaimed  Idle       0.000   504  0+01:23:40
C128000132.ip LINUX        INTEL  Unclaimed  Idle       0.000   504  0+02:33:39
C128000133.ip LINUX        INTEL  Unclaimed  Idle       0.000   504  0+00:53:39
C128000134.ip LINUX        INTEL  Unclaimed  Idle       0.000   504  0+00:03:39
C128000135.ip LINUX        INTEL  Unclaimed  Idle       0.000   504  0+00:58:39
C128000136.ip LINUX        INTEL  Unclaimed  Idle       0.040   504  0+02:03:42
C128000137.ip LINUX        INTEL  Unclaimed  Idle       0.000   504  0+01:08:42
C128000138.ip LINUX        INTEL  Unclaimed  Idle       0.000   504  0+00:58:42
C128000139.ip LINUX        INTEL  Unclaimed  Idle       0.000   504  0+03:08:42

                     Total Owner Claimed Unclaimed Matched Preempting Backfill

         INTEL/LINUX    87     0       8        79       0          0        0

               Total    87     0       8        79       0          0        0
griduser@C128001001:~$
```

Figure 5-1. Example run of condor_status.

5.1.3 SimpleNodes on Planet-Lab

Because of Brunet's ability to self-heal, even if all the SimpleNodes go offline, the system will reform and stay connected, although no new nodes will be able to connect. To ensure that a majority of Planet-Lab SimpleNodes are active, a script that checks the status of every node in the pool runs every other day stating if SimpleNode is on or not. In the event of an upgrade or finding nodes down, another script will update or reinstall SimpleNode on all the Planet-Lab nodes parallely.

5.1.4 Grid Appliance System Independence

A major goal for the Grid Appliance is true system independence, this is provided by the system in a file concept that can be run on x86 virtual machines and emulators, such as VMware and Qemu respectively. To date the Grid Appliance has successfully been run

Type	Occurrence	% of total
Runs	272918	100.00%
0% Loss	272398	99.81%
33% Loss	446	0.16%
66% Loss	74	0.03%
100% Loss	395	0.14%
100% for entire execution	0	0

Table 5-1. The ping test results dating from 02/24/07 to 03/14/07. The first for loss entries refer to the runs, while the final entry contains data for individual peer to peer communication for all runs.

on Microsoft Windows, Apple's MAC OS/X, and Linux systems supporting VMware. For effect, images of these test cases have been provided.

5.2 Performance Evaluation

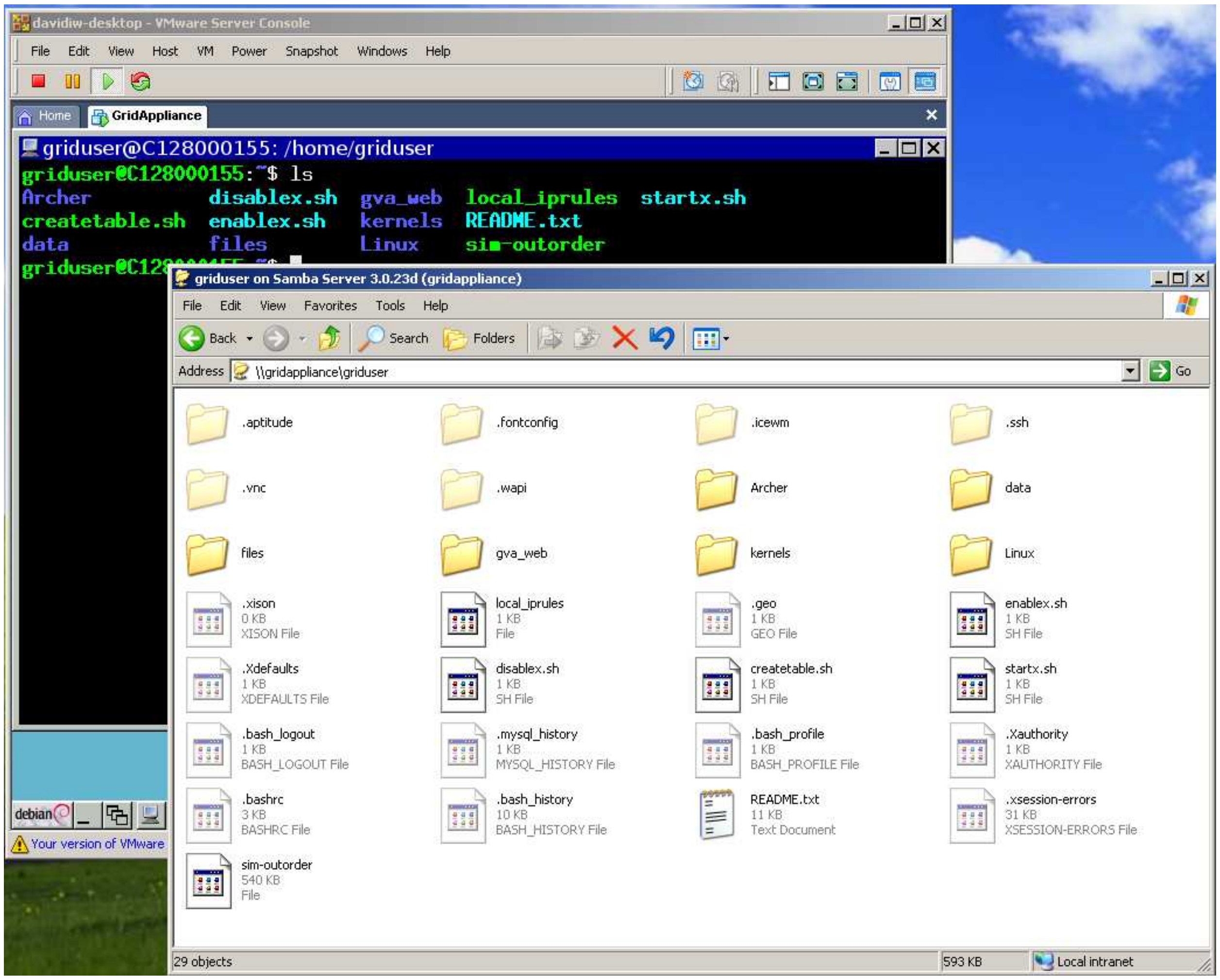
To evaluate the performance of the sandbox, three benchmarks were used. The benchmarks are

- SimpleScalar [59], a CPU-intensive computer architecture simulator which models the executions of applications with cycle-level accuracy[1]
- PostMark [61], a file system benchmark
- Iperf [62], a TCP throughput benchmark

The performance of these applications is evaluated with 3 different platforms: Grid Appliance as both VMware Server and Xen VMs, and Linux on the physical host. The purpose of these benchmarks is not to compare VMware to Xen or the physical hardware, but to investigate the cost of using virtual machines and networking for the appliance, with focus on machine configurations that would be expected in a desktop-Grid type environment. The host configuration was:

- a desktop-class system
- Pentium IV 1.7GHz CPU with 256KB on-chip cache
- 512MB RAM PC133 RAM
- PATA Hard drive at 66 MBps
- 100 Mbit Ethernet
- VMware Server 1.0.1
- Xen Testing Nightly Snapshot 09/26/06

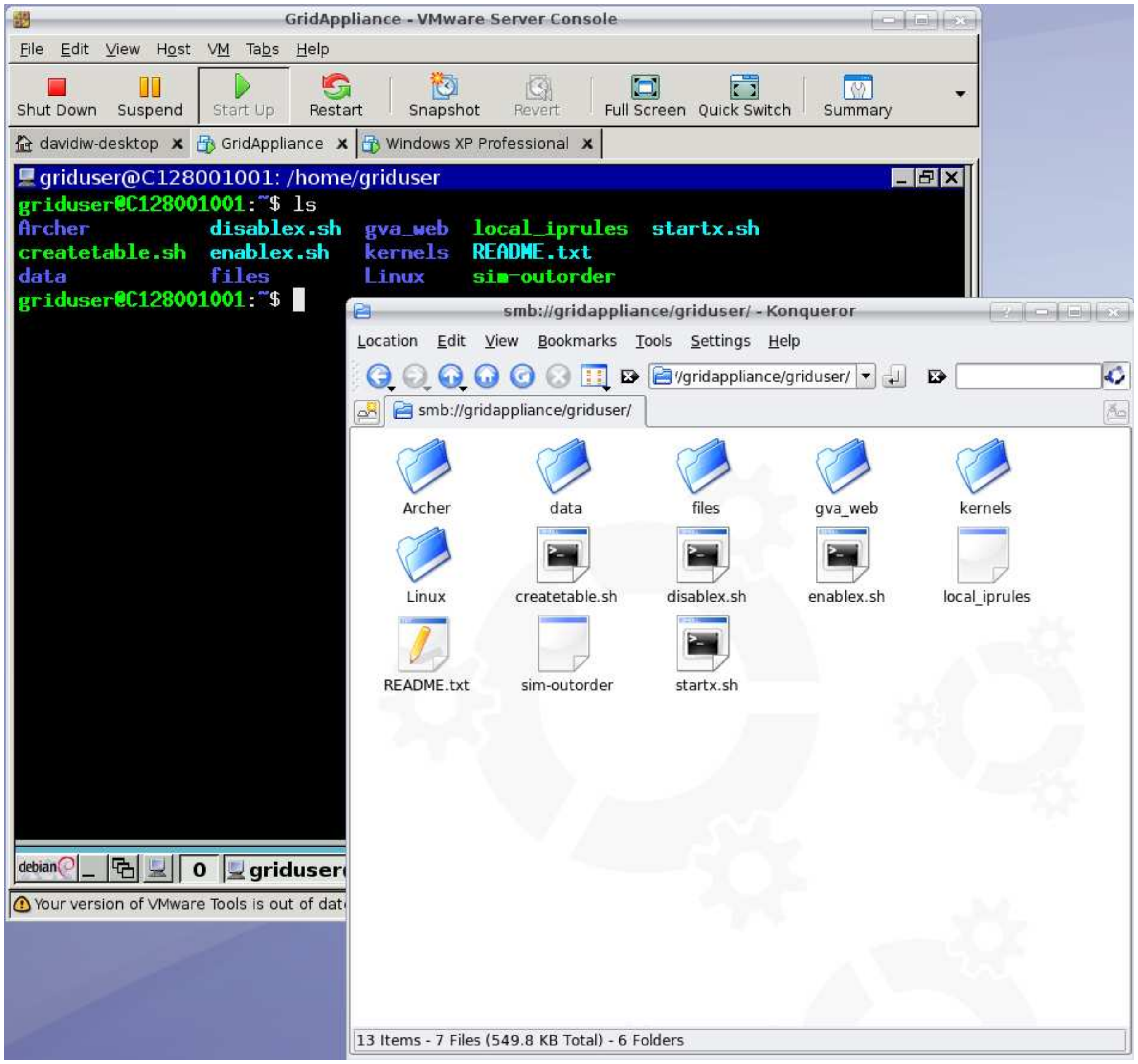
[1] the experiments employed the SPEC 2000s[60] Go benchmark


davidw-desktop - VMware Server Console
File Edit View Host VM Power Snapshot Windows Help
Home
GridAppliance
griduser@C128000155: /home/griduser
griduser on Samba Server 3.0.23d (gridappliance)
File Edit View Favorites Tools Help
Back
Search
Folders
Address
Go
debian
Your version of VMware
29 objects
593 KB
Local intranet

```
griduser@C128000155:~$ ls
Archer          disablex.sh  gva_web  local_iprules  startx.sh
createtable.sh  enablex.sh   kernels  README.txt
data            files        Linux    sim-outorder
griduser@C12
```

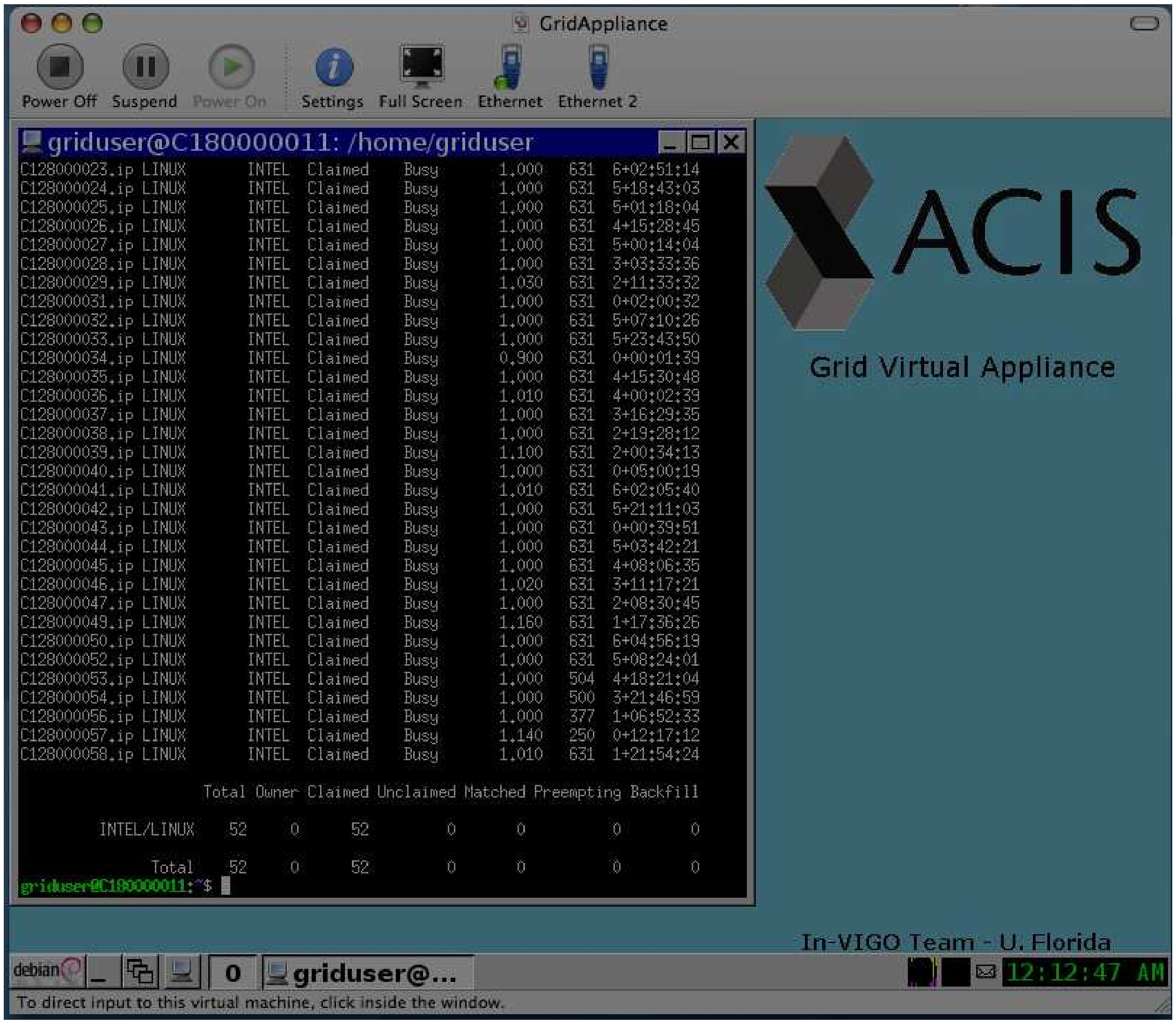
```
\\gridappliance\griduser

.aptitude     .fontconfig     .icewm        .ssh
.vnc          .wapi           Archer        data
files         gva_web         kernels       Linux
.xison        local_iprules   .geo          enablex.sh
0 KB          1 KB            1 KB          1 KB
XISON File    File            GEO File      SH File
.Xdefaults      disablex.sh   createtable.sh  startx.sh
1 KB            1 KB          1 KB            1 KB
XDEFAULTS File  SH File       SH File         SH File
.bash_logout       .mysql_history       .bash_profile       .Xauthority
1 KB               1 KB                 1 KB                1 KB
BASH_LOGOUT File   MYSQL_HISTORY File   BASH_PROFILE File   XAUTHORITY File
.bashrc       .bash_history       README.txt      .xsession-errors
3 KB          10 KB               11 KB           31 KB
BASHRC File   BASH_HISTORY File   Text Document   XSESSION-ERRORS File
sim-outorder
540 KB
File
```

Figure 5-2. Grid Appliance running on Windows using VMware Server.


GridAppliance - VMware Server Console
File Edit View Host VM Tabs Help
Shut Down Suspend Start Up Restart Snapshot Revert Full Screen Quick Switch Summary
davidiw-desktop GridAppliance Windows XP Professional
griduser@C128001001: /home/griduser
smb://gridappliance/griduser/ - Konqueror
Location Edit View Bookmarks Tools Settings Help
/gridappliance/griduser/
smb://gridappliance/griduser/
Archer data files gva_web kernels
Linux createtable.sh disablex.sh enablex.sh local_iprules
README.txt sim-outorder startx.sh
13 Items - 7 Files (549.8 KB Total) - 6 Folders
Your version of VMware Tools is out of dat
debian 0 griduser

```
griduser@C128001001:~$ ls
Archer          disablex.sh  gva_web  local_iprules  startx.sh
createtable.sh  enablex.sh   kernels  README.txt
data            files        Linux    sim-outorder
griduser@C128001001:~$
```

Figure 5-3. Grid Appliance running on Linux using VMware Server.



```
C128000023.ip LINUX      INTEL  Claimed    Busy       1.000   631  6+02:51:14
C128000024.ip LINUX      INTEL  Claimed    Busy       1.000   631  5+18:43:03
C128000025.ip LINUX      INTEL  Claimed    Busy       1.000   631  5+01:18:04
C128000026.ip LINUX      INTEL  Claimed    Busy       1.000   631  4+15:28:45
C128000027.ip LINUX      INTEL  Claimed    Busy       1.000   631  5+00:14:04
C128000028.ip LINUX      INTEL  Claimed    Busy       1.000   631  3+03:33:36
C128000029.ip LINUX      INTEL  Claimed    Busy       1.030   631  2+11:33:32
C128000031.ip LINUX      INTEL  Claimed    Busy       1.000   631  0+02:00:32
C128000032.ip LINUX      INTEL  Claimed    Busy       1.000   631  5+07:10:26
C128000033.ip LINUX      INTEL  Claimed    Busy       1.000   631  5+23:43:50
C128000034.ip LINUX      INTEL  Claimed    Busy       0.900   631  0+00:01:39
C128000035.ip LINUX      INTEL  Claimed    Busy       1.000   631  4+15:30:48
C128000036.ip LINUX      INTEL  Claimed    Busy       1.010   631  4+00:02:39
C128000037.ip LINUX      INTEL  Claimed    Busy       1.000   631  3+16:29:35
C128000038.ip LINUX      INTEL  Claimed    Busy       1.000   631  2+19:28:12
C128000039.ip LINUX      INTEL  Claimed    Busy       1.100   631  2+00:34:13
C128000040.ip LINUX      INTEL  Claimed    Busy       1.000   631  0+05:00:19
C128000041.ip LINUX      INTEL  Claimed    Busy       1.010   631  6+02:05:40
C128000042.ip LINUX      INTEL  Claimed    Busy       1.000   631  5+21:11:03
C128000043.ip LINUX      INTEL  Claimed    Busy       1.000   631  0+00:39:51
C128000044.ip LINUX      INTEL  Claimed    Busy       1.000   631  5+03:42:21
C128000045.ip LINUX      INTEL  Claimed    Busy       1.000   631  4+08:06:35
C128000046.ip LINUX      INTEL  Claimed    Busy       1.020   631  3+11:17:21
C128000047.ip LINUX      INTEL  Claimed    Busy       1.000   631  2+08:30:45
C128000049.ip LINUX      INTEL  Claimed    Busy       1.160   631  1+17:36:26
C128000050.ip LINUX      INTEL  Claimed    Busy       1.000   631  6+04:56:19
C128000052.ip LINUX      INTEL  Claimed    Busy       1.000   631  5+08:24:01
C128000053.ip LINUX      INTEL  Claimed    Busy       1.000   504  4+18:21:04
C128000054.ip LINUX      INTEL  Claimed    Busy       1.000   500  3+21:46:59
C128000056.ip LINUX      INTEL  Claimed    Busy       1.000   377  1+06:52:33
C128000057.ip LINUX      INTEL  Claimed    Busy       1.140   250  0+12:17:12
C128000058.ip LINUX      INTEL  Claimed    Busy       1.010   631  1+21:54:24

                     Total Owner Claimed Unclaimed Matched Preempting Backfill

         INTEL/LINUX    52     0      52         0       0          0        0

               Total    52     0      52         0       0          0        0
griduser@C180000011:~$
```

Figure 5-4. Grid Appliance running on MAC OS/X using VMware Fusion.

- Debian Etch for the physical host
- Debian Sarge for the Grid Appliance

The benchmarks were conducted on identically configured VMs, where only one uniprocessor virtual machine per physical CPU was deployed, and no other active processes running on the machines. A total of 256 MB of memory was given to each virtual machine. For networking tests, the VMs were run on two distinct, identically configured physical machines connected over 100 Mbit Ethernet.

5.2.1 SimpleScalar

SimpleScalar (version 3.0d) is used for benchmarking CPU performance. For the SimpleScalar test, SPEC 2000s Go was run using the alpha binaries found at [26]. The tests were executed over Condor. The parameters for go were "13 29 2stone9.in" (Figure 5-5. The SimpleScalar executable used was Sim-Cache. The results are consistent with previous findings; that is, virtual machines show low overhead in the case of processor intensive tasks (0.4% for Xen, and 10.6% for VMware.

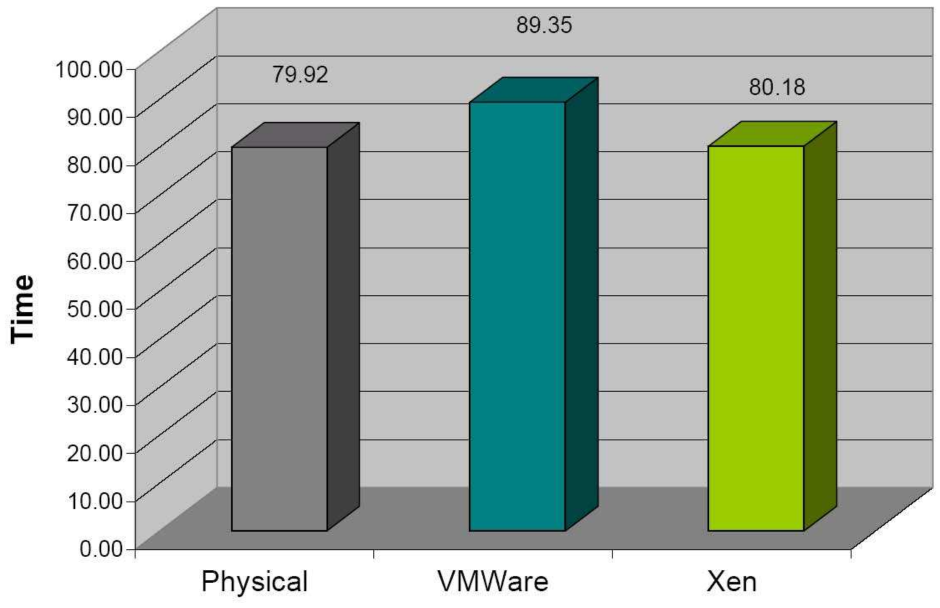


Figure 5-5. SimpleScalar results show the overall execution times (in minutes) for the execution of the Go benchmark in three different configurations.

5.2.2 PostMark

PostMark (version 1.51) is used for benchmarking disk performance, mainly for heavy I/O for many small files. For this test, the minimum and maximum size of files was 500 bytes up to 5,000,000 bytes (4.77 megabytes). To obtain steady state results, PostMark was configured with 5,000 file transactions. Results are shown in figure 5-6. The read / write ratio remains the same for all values.

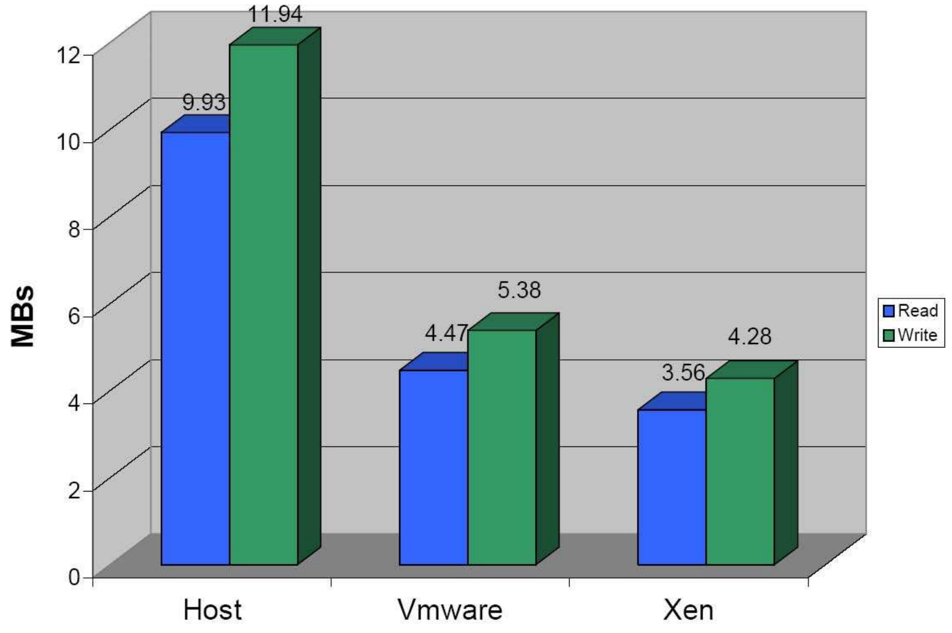


Figure 5-6. PostMark I/O throughput results, in read / write MB/s.

The use of file-backed VM drives and file system stacks greatly facilitate the deployment of the sandbox, but come with an associated performance cost. The measured performance of the sandbox with this I/O intensive benchmark is 55% to 64% of the physical host.

5.2.3 Iperf

Iperf is used to benchmark TCP network throughput. In this case, a 30 second transfer takes place and the throughput is measured at the end of this period. Iperf was run with the parameters -t 30. Given that the tests were conducted on 100 Megabit Ethernet, the results are not meant to suggest the sandbox results in a wide-area

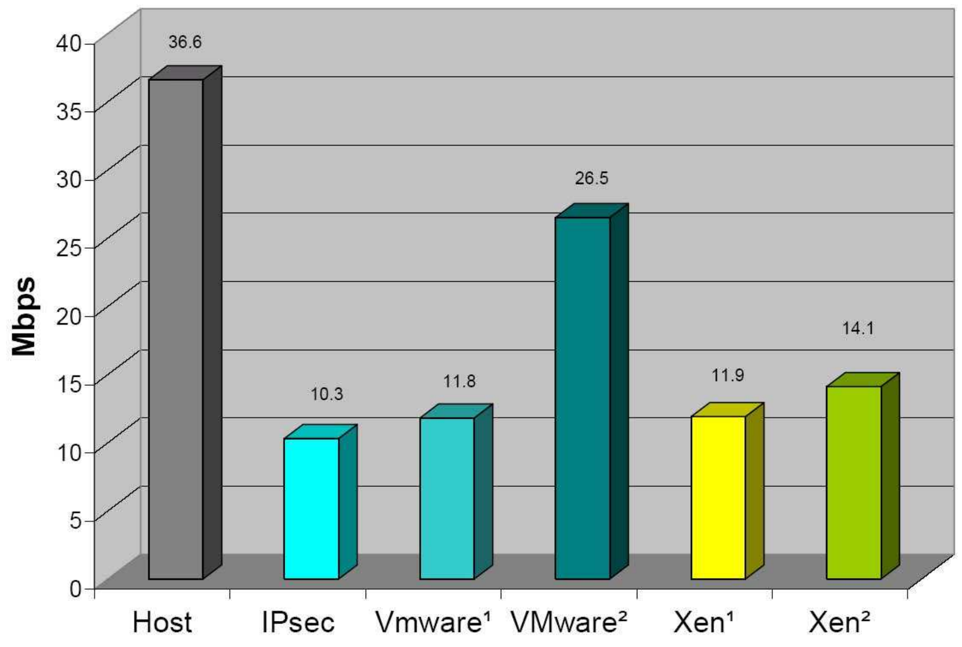


Figure 5-7. Iperf results are given in megabits per second, that is bigger is better.

environment, but to show the expected peak bandwidth of the sandbox configured with the IPOP user-level virtual networking software. The results establish that the VMware implementation where the virtual networking runs on the guest VM, without IPSec, delivers a bandwidth of 11.8 Mbps, whereas with IPSec the bandwidth is decreased to 10.3 Mbps. When the IPOP virtual network runs on the host, the virtual network bandwidth is improved substantially to 26.5 Mbps. This can be explained by the fact that the IPOP software is network-intensive; when it runs on the host, it is not subject to the virtualization overhead during its execution and can deliver better performance. Xen delivers 11.9 Mbps with IPOP on domU, and 14.1 Mbps with IPOP on dom0[2] . Results are shown in figure 5-7 with superscript 1 implying virtual networking inside the VM, where as 2 means virtual networking on the host.

[2] As of this writing, Xen responds with a warning message stating that negative segmentation is not supported when running IPOP. It is conceivable that the mono runtime environment uses negative segments and performance may be degraded due to this fact.

5.2.4 Discussion

The experimental results show a small overhead of the sandbox for compute-intensive applications, a conclusion that has been also observed in previous work[3, 42]. A substantial source of overheads in network throughput performance comes from the IPOP virtual network implementation, which currently is entirely in user space. Nonetheless, the sandbox network throughput performance levels are acceptable for the intended application of this sandbox for compute-intensive applications and in wide-area environments. Running the IPOP software on the host allows for stronger isolation of the virtual network traffic because the process that captures and tunnels packets resides on a separate domain from the compute sandbox. Furthermore, virtual networking on the host provides the best observed throughput. However, it comes with the downside of requiring a user to install additional software. An alternative that does not require IPOP to run on the host but still provides strong virtual network isolation is to add a second level of virtualization (e.g., by running the Xen appliance within a VMware hosted I/O environment). This is the subject of on-going investigations.

CHAPTER 6
CONCLUSION

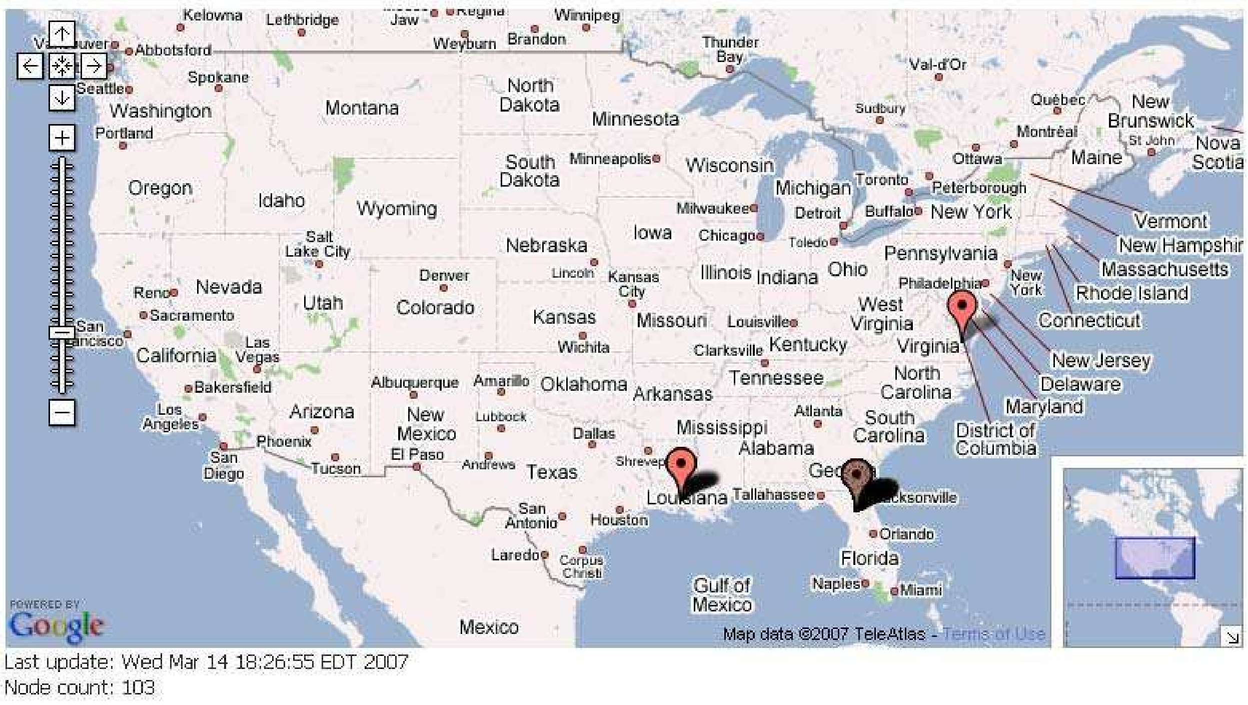
This chapter discusses current deployments of the Grid Appliance, future work, and a brief conclusion.

6.1 Current Deployments

The Grid Appliance is currently being used by different application domains. A Condor pool is available to the nanoHUB for the execution of batch jobs, and customizations are underway to enable an application development environment intended to foster the addition of graphical, interactive applications to the nanoHUB cyber-infrastructure by its community.

Grid Appliances are also being deployed in support of hurricane storm surge models as part of the SCOOP [34] project. In this context, they are used in two different ways: on-line, dynamic data-driven execution of models, and off-line retrospective analysis. In the event-driven scenario, the computation on appliances are triggered by data streams made available by sources such as the National Hurricane Center, and model simulation results are published to the SCOOP community through SCOOPs data transport system (UniData's LDM [63]). Event-triggered jobs can be scheduled to run locally on individual appliances, or submitted to other nodes in the pool through Condor. In the retrospective analysis scenario, data is retrieved from the SCOOP archive, simulation model executions are dispatched to a Condor pool, and results are published back to the SCOOP archive through LDM. Currently, a total of 32 appliances serving these purposes have been deployed at four SCOOP sites.

Another usage scenario where Grid Appliances are being developed in the domain of coastal sciences is an appliance for education and training of researchers, students, and the public at large in cyber-infrastructure techniques. In this usage scenario, the appliance integrates surge simulation models, Condor middleware, visualization software, as well as tutorials and educational material on cyber-infrastructure and coastal and estuarine science. As a result, it enables end-to-end usage, including application development, data



Last update: Wed Mar 14 18:26:55 EDT 2007
Node count: 103

Figure 6-1. Current deployment of Grid Appliances from the Grid Appliance Web Interface.

input, simulation execution, data post-processing and visualization. Users who are not familiar with Grid computing can download and install an appliance, and submit a sample simulation from their own home or office computer to other Grid Applianc es all within minutes. In contrast, configuring physical machines to run the appropriate middleware and simulation models takes a level of familiarity with installing and configuring various software and middleware packages that is a significant barrier to adoption by many scientists, engineers and students.

There is also a general use deployment containing over 90 nodes. The web interface monitors and submits jobs to this pool and has a picture of the United States map showing the current location of different nodes (Figure 6-1). The web site is also used to monitor the other Grid Appliances pools. The website is available at http://wow.acis.ufl.edu.

6.2 Conclusion

Over nearly a year ago, the Grid Appliance was only a virtual machine with Condor and an unmanageable IPOP weighing over 2 gigabytes. Since then significant features such as a graphical user interface, network file access by Samba, DHT over DHCP capable IPOP using standard operating system DHCP, an interactive web interface, stackable file systems, and user-friendliness for reconfigurability have been added. The Grid Appliance provides grid capabilities for all technology backgrounds having both console and web interfaces. Most importantly, the Grid Appliance has low overhead for tasks that it was designed to run. While there may be other solutions for virtual grid workstations, a related works review establishes that none have the same depth that Grid Appliance provides.

Even with all these features there is still much room for future research in the Grid Appliance. A few focal points for research include a BitTorrent based file system, distributed Condor, nested virtualization for superior sandboxing, and tools for rapid development of simple interactive web pages.

REFERENCES

[1] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1160055

[2] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *Int. J. High Perform. Comput. Appl.*, vol. 15, no. 3, pp. 200–222, August 2001. [Online]. Available: http://portal.acm.org/citation.cfm?id=1080667

[3] R. J. Figueiredo, P. A. Dinda, and J. A. B. Fortes, "A case for grid computing on virtual machines," in *ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems.* Washington, DC, USA: IEEE Computer Society, 2003. [Online]. Available: http://portal.acm.org/citation.cfm?id=851917

[4] VMware. (2007, March) Vmware workstation, vmware server, vmware player, vmware esx. [Online]. Available: http://www.vmware.com

[5] 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 *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles.* New York, NY, USA: ACM Press, 2003, pp. 164–177. [Online]. Available: http://portal.acm.org/citation.cfm?id=945462

[6] InnoTek. (2007, March) Virtualbox. [Online]. Available: http://www.virtualbox.org

[7] Parallels. (2007, March) Parallels workstation, parallels desktop. [Online]. Available: http://www.parallels.com

[8] F. Bellard. (2007, March) Qemu. [Online]. Available: http://fabrice.bellard.free.fr/qemu/

[9] Qumranet. (2007, March) Kernel-based virtual machine for linux. [Online]. Available: http://kvm.qumranet.com/kvmwiki

[10] D. I. Wolinsky, A. Agrawal, P. O. Boykin, J. Davis, A. Ganguly, V. Paramygin, P. Sheng, and R. J. Figueiredo, "On the design of virtual machine sandboxes for distributed computing in wide area overlays of virtual workstations," in *First Workshop on Virtualization Technologies in Distributed Computing (VTDC)*, November 2006.

[11] C. S. Yeo, R. Buyya, H. Pourreza, R. Eskicioglu, P. Graham, and F. Sommers, "Cluster computing: High-performance, high-availability, and high-throughput processing on a network of computers," in *Handbook of Nature-Inspired and Innovative Computing: Integrating Classical Models with Emerging Technologies*, A. Y. Zomaya, Ed. New York, NY: Springer, 2006, ch. 16, pp. 521–551.

[12] L. Peterson and D. Culler. (2007, March) Planet-lab. [Online]. Available: http://www.planet-lab.org

[13] Altair. (2007, March) Pbs pro. [Online]. Available: http://www.altair.com/software/pbspro.htm

[14] ——. (2007, March) Openpbs. [Online]. Available: http://www.openpbs.org/

[15] C. Resources. (2007, March) Torque resource manager. [Online]. Available: http://www.clusterresources.com/pages/products/torque-resource-manager.php

[16] M. Livny, J. Basney, R. Raman, and T. Tannenbaum, "Mechanisms for high throughput computing," *SPEEDUP Journal*, vol. 11, no. 1, June 1997.

[17] J. Basney and M. Livny, "Deploying a high throughput computing cluster," in *High Performance Cluster Computing: Architectures and Systems, Volume 1*, R. Buyya, Ed. Prentice Hall PTR, 1999.

[18] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the condor experience." *Concurrency - Practice and Experience*, vol. 17, no. 2-4, pp. 323–356, 2005.

[19] Sun. (2007, March) gridengine. [Online]. Available: http://gridengine.sunsource.net/

[20] P. Computing. (2007, March) Load sharing facility, lsf. [Online]. Available: http://www.platform.com/

[21] A. Staicu, J. Radzikowski, K. Gaj, N. Alexandridis, and T. El-Ghazawi, "Effective use of networked reconfigurable resources," in *Military Applications of Programmable Logic Devices*, 2001.

[22] R. Vogelbacher. (2007, March) Job scheduler/resource manager evaluation. [Online]. Available: http://gridweb.cti.depaul.edu/twiki/bin/view/IBG/SchedulerProductEvaluation

[23] G. Alliance. (2007, March) Globus toolkit. [Online]. Available: http://www.globus.org/toolkit/

[24] G. J. Popek and R. P. Goldberg, "Formal requirements for virtualizable third generation architectures," in *SOSP '73: Proceedings of the fourth ACM symposium on Operating system principles*, vol. 7, no. 4. New York, NY, USA: ACM Press, October 1973. [Online]. Available: http://portal.acm.org/citation.cfm?id=808061

[25] V. Bala, E. Duesterwald, and S. Banerjia, "Dynamo: a transparent dynamic optimization system," in *PLDI '00: Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation*, vol. 35, no. 5. ACM Press, May 2000, pp. 1–12. [Online]. Available: http://portal.acm.org/citation.cfm?id=349303

[26] K. Adams and O. Agesen, "A comparison of software and hardware techniques for x86 virtualization," in *ASPLOS-XII: Proceedings of the 12th international conference on*

Architectural support for programming languages and operating systems. New York, NY, USA: ACM Press, 2006, pp. 2–13.

[27] Cisco. (2007, March) Cisco vpn. [Online]. Available: http://www.cisco.com/en/US/products/sw/secursw/ps2308/index.html

[28] J. Yonan. (2007, March) Openvpn. [Online]. Available: http://openvpn.net/

[29] M. Tsugawa, A. Matsunaga, and J. A. B. Fortes, "Virtualization technologies in transnational dg," in *dg.o '06: Proceedings of the 2006 international conference on Digital government research.* New York, NY, USA: ACM Press, 2006, pp. 456–457.

[30] A. Sundararaj and P. Dinda, "Towards virtual networks for virtual machine grid computing," 2004. [Online]. Available: http://citeseer.ist.psu.edu/645578.html

[31] I. X. Jiang, "Violin: Virtual internetworking on overlay." [Online]. Available: http://citeseer.ist.psu.edu/714412.html

[32] C. P. Wright and E. Zadok, "Unionfs: Bringing file systems together," *Linux Journal*, no. 128, pp. 24–29, December 2004.

[33] A. Ganguly, A. Agrawal, O. P. Boykin, and R. Figueiredo, "Ip over p2p: Enabling self-configuring virtual ip networks for grid computing," in *Proceedings of International Parallel and Distributed Processing Symposium (IPDPS). To appear*, Apr 2006.

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

[35] M. Krasnyansky. (2007, March) Universal tun/tap device driver. [Online]. Available: http://www.kernel.org/pub/linux/kernel/people/marcelo/linux-2.4/Documentation/networking/tuntap.txt

[36] R. Droms, *RFC 2131 Dynamic Host Configuration Protocol*, March 1997.

[37] R. D. S. Alexander, *RFC 2132 DHCP Options and BOOTP Vendor Extensions.*

[38] P. Mockapetris, *RFC 1034 Domain names - concepts and facilities*, November 1987.

[39] ——, *RFC 1035 Domain names - implementation and specification*, November 1987.

[40] D. C. Plummer, *RFC 0826 Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware*, November 1982.

[41] A. Ganguly, D. I. Wolinsky, P. O. Boykin, and R. J. Figueiredo, "Decentralized dynamic host configuration in wide-area overlay networks of virtual workstations," in *WORKSHOP ON LARGE-SCALE, VOLATILE DESKTOP GRIDS*, March 2007.

[42] S. Santhanam, P. Elango, A. A. Dusseau, and M. Livny, "Deploying virtual machines as sandboxes for the grid," in *WORLDS*, 2005.

[43] N. Andrade, L. Costa, G. Germoglio, and W. Cirne, "Peer-to-peer grid computing with the ourgrid community," May 2005.

[44] S. Kent, *RFC 2401 Security Architecture for the Internet Protocol*, November 1998.

[45] S. Adabala, V. Chadha, P. Chawla, R. Figueiredo, J. Fortes, I. Krsul, A. Matsunaga, M. Tsugawa, J. Zhang, M. Zhao, L. Zhu, and X. Zhu, "From virtualized resources to virtual computing grids: the in-vigo system," *Future Gener. Comput. Syst.*, vol. 21, no. 6, pp. 896–909, 2005.

[46] N. F. C. Nanotechnology. (2007, March) nanohub. [Online]. Available: http://www.nanohub.org/about/

[47] E. Roberts, M. Dahan, J. Boisseau, and P. Hurley. (2007, March) Teragrid user portal. [Online]. Available: https://portal.teragrid.org/gridsphere/gridsphere

[48] (2007, March) Condorview. [Online]. Available: http://www.cs.wisc.edu/condor/manual/v6.7/3_4Contrib_Module.html

[49] A. Harter and T. Richardson. (2007, March) Virtual network computing. [Online]. Available: http://www.cl.cam.ac.uk/research/dtg/attarchive/vnc/index.html

[50] M. McLennan, D. Kearney, W. Qiao, D. Ebert, and C. S. R. Kent Smith. (2007, March) Rappture toolkit. [Online]. Available: https://developer.nanohub.org/projects/rappture/

[51] J. J. Garret. (2005, February) Ajax: A new approach to web applications. [Online]. Available: http://www.adaptivepath.com/publications/essays/archives/000385.php

[52] H. Edlund. (2007, March) Drall. [Online]. Available: http://home.gna.org/drall/

[53] Y. Goland, E. Whitehead, A. Faizi, S. Carter, and D. Jensen, *RFC 2518 HTTP Extensions for Distributed Authoring – WEBDAV*, February 1999. [Online]. Available: http://dav.sourceforge.net/

[54] K. Keahey, I. Foster, T. Freeman, X. Zhang, and D. Galron, "Virtual workspaces in the grid," in *Proceedings of Europar 2005*, September 2006.

[55] K. Keahey, K. Doering, and I. Foster, "From sandbox to playground: Dynamic virtual environments in the grid," in *Proceedings of 5th International Workshop in Grid Computing*, November 2004.

[56] R. Andersen and B. Vinter, "Minimum intrusion grid- the simple model," in *14th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2005)*, June 2005.

[57] B. University of California. (2007, March) Seti@home. [Online]. Available: http://setiathome.berkeley.edu/

[58] V. Pande and Stanford. (2007, March) Folding@home distributed computing. [Online]. Available: http://folding.stanford.edu/

[59] S. LLC. (2007, March) Simplescalar 3.0d. [Online]. Available: http://www.simplescalar.com

[60] S. P. E. Corporation. (2007, March) Spec cpu 2000. [Online]. Available: http://www.spec.org/cpu

[61] J. Katcher, "Postmark: a new file system benchmark," in *Network Appliance*, October 1997.

[62] T. B. of Trustees of the University of Illinois. (2007, March) Iperf. [Online]. Available: http://dast.nlanr.net/projects/iperf/

[63] G. P. Davis and R. K. Rew, "The unidata ldm: Programs and protocols for flexible processing of data products," in *Tenth International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology*, October 1996, pp. 131–136.

BIOGRAPHICAL SKETCH

David Wolinsky was born October 31, 1982. He attended the University of Florida for a 4-year Bachelor of Science degree in Computer Engineering followed by a 2-year Master of Science degree in electrical engineering, respectively. The last 2 years of his life have been consumed by virtual machines.