On the Design of Virtual Machine Sandboxes for Distributed Computing in Wide-area Overlays of Virtual Workstations

David Isaac Wolinsky†, Abhishek Agrawal†, P. Oscar Boykin†, Justin R. Davis‡, Arijit Ganguly†, Vladimir Paramygin‡, Y. Peter Sheng‡, Renato J. Figueiredo†

†Advanced Computing and Information Systems, University of Florida, ‡Civil and Coastal Engineering, University of Florida

***Abstract*— With recent advances in virtual computing and the revelation that compute-intensive tasks run well on system virtual machines (VMs), the ability to develop, deploy, and manage distributed systems has been ameliorated. This paper explores the design space of VM-based sandboxes where the following techniques that facilitate the deployment of secure nodes in Wide-area Overlays of virtual Workstations (WOWs) are employed: DHCP-based virtual IP address allocation, self-configuring virtual networks supporting peer-to-peer NAT traversal, stacked file systems, and IPsec-based host authentication and end-to-end encryption of communication channels. 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.**

***Index Terms*— Grid Computing, Virtual Computing, Virtual Machines, Virtual Networking.**

I. Introduction

Virtualization techniques can greatly facilitate the deployment of wide-area distributed computing infrastructures with low administrative overheads while keeping resources secure. Virtual machines (VMs) assist in the deployment of compute nodes, because of their decoupling from the operating system running on the physical machine [1] and ability to package entire virtual disks as regular files stored on a host. Virtual networks facilitate the interconnection of VMs deployed across different network domains, particularly in the presence of firewall and NAT (Network Address Translation) routers, by decoupling the virtual IP address space from the underlying physical network. Leveraging on these capabilities, previous studies have motivated the deployment of virtualized infrastructures for wide-area computing [6,27,28,29], including self-organizing **W**ide-area **O**verlays of virtual **W**orkstations [5] (WOWs). This paper explores the design space of VM-based sandboxes enabling self-configurable, scalable, secure deployment of computing nodes in a WOW.

Because of the extensive use of virtualization, the techniques described in this paper reuse and integrate various software components such as VM monitors (VMMs), DHCP clients, IPsec, and UnionFS stacks. The novelty of this approach lies in how the system is architected and integrated to leverage the encapsulating capabilities of system VMs coupled with the homogeneous virtual IP address space provided by virtual networks to integrate all such software in an easy to deploy VM-based distributed system. This is achieved by requiring only a single software package to be installed on the VM's host (a free VMM) and a single VM image. Furthermore, no physical IP address needs to be allocated by the host site – the VM can use a NAT-based virtual network interface.

Specifically, this paper describes and evaluates approaches to: 1) support host and network isolation by means of machine virtualization, network tunneling and network traffic filtering; 2) support site-specific customization and system upgrades through the use of stacked file systems; 3) automatically provide virtual IP addresses and names to WOW nodes running unmodified O/Ss by means of virtualized DHCP and DNS servers; and 4) support end-to-end payload encryption and public-key based host authentication at the virtual IP layer by enabling scalable deployment of X.509-based authentication and host-to-host IPsec transport mode.

These techniques can be used in supporting collaborative environments where resources from individual participants from a community can be pooled together with very little administrative overheads. The VM image of a WOW node is defined once by a system administrator who is mindful of the configuration needs of the community, and then can be instantiated in many hosts across the wide area to form a homogeneously configured virtual infrastructure. Examples of collaborative environments where these virtualization techniques are currently being applied include user-contributed Condor nodes in the nanoHUB cyberinfrastructure for nano-electronics simulation [33] and a virtual cluster to run forecast and hindcast storm surge models in the SURA Coastal Ocean Observing and Prediction (SCOOP) Program [34].

These techniques have been successfully applied in the design of a VM-based "appliance" for grid computing termed throughout this paper as WOW sandbox (appliance), which allows the dynamic creation of Condor-based [26] pools for wide-area high-throughput computing. The appliance employs virtual machines in the form of VMware [12] and Xen [4] and virtual networking through IPOP [2, 5]. Through the use of free VMMs such as VMware's Player or Server or XenSource's Xen, deployment of a fully configured WOW sandbox appliance is as simple as setting up a VMM and opening a file[1]; once instantiated, the virtual machine acquires

[1] A VM image with the current release of a WOW sandbox is available for public download and use from http://www.acis.ufl.edu/~ipop/grid

a virtual IP address and automatically connects itself to the virtual network to become a submit/execute node of a Condor pool.

Quantitative experiments are performed to analyze the performance of the WOW sandbox appliance for workloads that exercise CPU, disk and network I/O subsystems: SimpleScalar (a computer architecture simulator), PostMark (a file system benchmark) and IPERF (an application to measure TCP network throughput). Appliance configurations where the virtual networking software runs on either the VM host or the guest are studied. Results show that the slowdown due to virtualization for the CPU-intensive workload is at most 10.6%. For the disk I/O benchmark, the overall observed virtualization overhead ranges from 55% to 64%. Results for the network benchmark show that the virtualized network interface delivers TCP throughput ranging from 10.3Mbps to 26.5Mbps, depending on the VMM and virtual networking configurations.

This paper is organized as follows. Section II discusses the architecture of the system, and Section III discusses security techniques. Section IV details current distributions of the WOW sandbox. Section V presents results and analyses of the quantitative performance experiments. Section VI discusses related work, and Section VII concludes the paper and presents directions for future work.

II. System Overview

Virtualization techniques expedite the deployment of distributed systems by enabling systems with homogeneous configurations to be instantiated on resources that are heterogeneously configured. Previous work has presented a case for the deployment of virtualized wide-area overlay networks of workstations, where heterogeneity in the O/S configuration of hosts and in the configuration of IP address spaces and NAT/firewall devices at different sites are dealt with by the virtualization layer [5]. Related approaches based on the use of system VMs and different virtual networking techniques are discussed in [27, 28, 29].

This paper considers the design of VM sandboxes that facilitate the deployment of wide-area collaborative environments. The target usage is a scenario where the VMs are developed, tuned and configured by administrators of the collaborative environment, and deployed by a potentially large number of users. This model of deployment in distributed systems has been successfully applied in projects including PlanetLab [32], where system configurations tailored to wide-area networking experimentation are prepared and disseminated by "PlanetLab Central", and the Open Science Grid [31], where system configurations tailored to grid computing are provided to system administrators as reference implementations with an easy to install collection of common services. These systems, however, do not make use of a virtualized network interconnecting the nodes of the distributed system. This paper focuses on systems where both nodes and network interconnections are virtualized.

The approach analyzed in this paper is based on the design of sandboxes as VM "appliances" [32] which a) encapsulate all the software configuration of a node into a single, easy to disseminate VM image, and b) self-configure its virtual IP address on an overlay network upon instantiation, building upon the peer-to-peer discovery, routing and NAT-traversal techniques implemented in IPOP [2]. The following subsections describe the different components of such sandboxes.

A. Virtual Machines

The creation of sandboxes [3, 8] and the ease of deployment images are key features for virtual machines being the base of this design. Prior to discussing these two points, it is important to state that in the case of running computationally intensive jobs virtual machines perform nearly identical to physical machines [1]. Furthermore, in the case of wide-area systems, the overheads associated with network virtualization are sufficient for applications such as high-throughput distributed computing [5].

Two VM-based sandboxing techniques are considered; the first relies on a hosted I/O VMM (currently VMware) where both the IPOP virtual networking software and WOW users run in the same "guest" O/S; the second is based on running IPOP and jobs from remote users on distinct "domains" (e.g. domU and dom0 in Xen, guest and host in VMware). The former case is the simplest to deploy, requiring only a hosted I/O VMM on the host – no additional software on the host is needed. The latter is motivated by virtual networking isolation and security issues which are discussed in detail in Section III. The common features in both cases include the use of file based disk images for hard drives or file systems. VMs are assumed to have a connection to the Internet, and the virtual networking techniques in use allow VMs to join IP overlay networks even if they are behind one or more levels of NATs [2, 6, 13].

B. Virtual Network

In order to create a virtual private network capable of self-configuring NAT traversal, this design employs the IPOP (IP-over-P2P) overlay [2]. IPOP is deployed as a user-level application that reads and writes packets to a virtual Ethernet device ("tap" [10]) and sends and receives them over a peer-to-peer virtual network based upon Brunet [11]. From a user's perspective, IPOP acts as any other networking device, so its embedding into the sandbox requires no extra configuration from a user of the system. Both IPOP and Brunet are coded in C#; runtime environments for this language are available for various Unix platforms and Windows; currently IPOP runs in Mono for Linux.

With the goals of facilitating the configuration of sandboxes and the integration with a variety of O/Ss, the ability to handle DHCP [14, 15], ARP [20], and DNS [16, 17] has been added to the baseline IPOP design. DHCP supports simple configuration of IP addresses and integration with existing DHCP clients that are ubiquitous in modern O/S distributions. ARP handling complements the DHCP support by automating

This WOW sandbox self-configures a Condor submit/execute node and connects to a pool of resources upon instantiation.

the configuration of the virtual MAC address of the IPOP gateway and of the routing tables for the virtual IP address space routed by IPOP. DNS allows supporting applications that require name resolution [7].

DHCP provides the ability for dynamic configuration of virtual IPs for IPOP. Currently this is achieved by specifying, in the IPOP configuration file, the address of a virtualized DHCP server[2]. The VM runs a DHCP client on the TAP device; IPOP captures DHCP client requests from the tap device and forwards them to the DHCP server, which then responds with a virtual IP address and lease information. The implementation follows the DHCP protocols [14, 15] and handles ARP requests automatically, so that IPOP should be able to run on any operating system that supports DHCP and a C# runtime environment.

Many programs, including Condor [13], require the use of host names, and therefore some means of resolving host names to virtual IP addresses on IPOP. While implementations based on the use of hostname files suffice for many applications where names are statically bound to IP addresses (e.g. in many private networks within a cluster), they do not scale well for large numbers of hosts. For example, tests were run with 'hosts' files containing millions of entries with lookup times taking up to a few seconds. Instead, a DNS server has been implemented in Python [24] which can be run either on a remote server (important if name-IP mappings change over time), or locally on the client, (for low-latency name resolution in the case of static name-IP mappings).

To keep administration simple, a script has been added to the sandbox to automatically update IPOP. The script checks for the latest version information from a server, which is then compared against a local version for an update decision. During an update, the sandbox downloads IPOP, moves it into the location of the local version, and restarts the IPOP service. This is perceived as a temporary loss of network connectivity, which is typically within the timeframe where the TCP protocol is able to maintain established connections [5].

C. *Modules*

To achieve the goal of easy deployment, it is important that the sandbox design supports updates (e.g. security patches, bug fixes, new features) of existing VMs and user customizations required to cater to specific needs of a user community. This issue is addressed through the use of a stacked file system based on UnionFS [9]. The stacked file system design (Figure 1) allow changes to the base VM configurations to be easily merged with local changes to the file system of already-deployed appliances, and allows domain-specific modules to be developed independently from the base image.

The stacked file system design enables two models of use: 'develop' and 'release'. In 'develop', a developer adds domain-specific features to the VM image, such as particular libraries or packages of interest to a user community. The results of the developer's work will be found in the "module" image, which the developer can distribute to any end users. End users will run the machine in 'release' mode, which employs the module, but stores user files in another image. Furthermore, it is possible to update the underlying "base" image by simply replacing the existing disk image without loss of any data stored in upper-level stacks.

A stackable file system supports the merging of many read-only and read-write file systems together into one file system. If there are one or more read-write file systems, the top most file system is where all the changes reside. If only the top most file system were cleared, independently of the stack, then the stacked file system would revert to its original form.

In development mode, only two file systems are used in the stack: "base" as read-only, and "module" as read-write. The base is the Grid Appliance's base files and on first use, the module is an empty file system. The union is made at root '/', therefore any changes to the system will be reflected in the module file system. After a user has configured the module, they must run a script which will remove any files which have been predetermined to not belong on a released clean virtual machine, such as '/var/log' and '/usr/local/ipop/var'. The user is responsible for deleting any files not in this directory that should not be released. After which, the user can distribute their module file system to other users. These users would place this in the same directory as their sandbox and upon booting would have the functionality provided by the module.

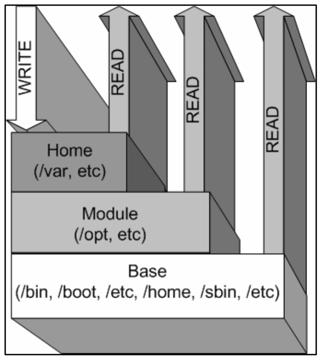

Fig. 1. Example of UnionFS file system stack showing the WOW appliance. In release mode, files can be read from any of the three layers, but all writes are sent to only the top file system. The folders read to or written from are not limited to those listed in parentheses.

This approach allows the creation of a generic baseline image that does not contain any large, domain-specific application but allows for easy addition of such tools. The choice of modules added to the baseline image is often dependent on specific needs of users of an application domain. For example, in coastal ocean modeling applications developed for the SCOOP project there is often the need for a variety of Perl scripts which require several modules not found in default configuration, as well as packages to handle GIS (Geographic Information System) data and visualization (e.g. Google Earth). The nanoHUB collaborative environment has a different set of requirements, including the Rappture Tcl/Tk-based GUI toolkit and a WebDAV file system client. These domain-specific configurations can be kept in separate stacked file system modules, which help both kee

[2] It is also possible to deploy a decentralized DHCP server by using a distributed hash table deployed across IPOP nodes, a feature which is under on-going implementation in IPOP.

baseline WOW sandbox appliance as small as possible, and greatly expediting the release of updated appliance baseline images, since modules should not need to be updated or re-released.

In release mode, three file systems are stacked: "base", "module", and "home" with the last being the only read-write. With this system, a user can easily erase all data accumulated while running in release mode by simply erasing the home file system. This can be used for security purposes, if a user wishes to copy a used appliance but does not want to go through the process of re-downloading software. In the base distribution of the WOW appliance, there are scripts to facilitate the management of modules, with commands to replace either their module or home file system. Fig. 1 presents a working example of how 'release' mode works.

III. Security

An important concern in the design of WOW sandboxes are the potential security risks that may arise when a site hosts a WOW node. The overall design approach to securing WOW sandboxes is based on a combination of techniques to enforce resource isolation and authentication among virtual hosts.

Isolation of the WOW node from the underlying physical machine is provided by the VM monitor. Isolation of WOW virtual network traffic from the physical network is provided via tunneling by IPOP. The approach which delivers strongest isolation is to run IPOP in a different domain (e.g. dom0 in Xen, or the host in VMware) from the WOW node which is accessible by remote users (domU in Xen, guest in VMware, Figure 2). These are discussed in subsections III.A and III.B.

In addition to host and network isolation, sites hosting sandboxes may also require that only authenticated nodes can join a given network. To this end, WOW appliances can be deployed with a mechanism to enforce IP-layer authentication based on X.509 certificates. Such host certificates can be signed by an authority which oversees the administration of a given WOW deployment. With IPsec, VM authentication and end-to-end traffic encryption can be provided, which is key to supporting many existing applications developed for LANs (e.g. network file systems) on WOWs. Subsection III.C describes the IPsec-based WOW sandbox setup in detail.

A. Virtual network tunneling on host

VMs such as Xen and VMware provide sandboxing mechanisms to isolate guest from host execution domains [3]. The first approach explored confines the sandbox network traffic to a virtual network by running the tunneling software on a separate domain, In this scenario, the remote user is contained in a VMware guest (or Xen domU), while IPOP runs in the VMware host (or Xen dom0'). A bridge on the host allows the IPOP process to capture packets from the VM's virtual network interface and to tunnel them through the physical network interface.

With this in place, all traffic generated by a user of the WOW sandboxed is completely isolated from the physical network by means of tunneling, and the network interface presented to the sandbox is only routable on the virtual IP address space. Because the IPOP process runs on a separate execution domain, even if a remote user were to escalate privileges within the VM, their network traffic would still be confined to the WOW network; hence, users would not be able to launch attacks on physical machines in the hosting site or public Internet hosts from the WOW node. Communication between virtual and physical machines (e.g. for file transfers) can be achieved, by establishing host-only interfaces which are firewalled to enable only specific protocols, such as SSH.

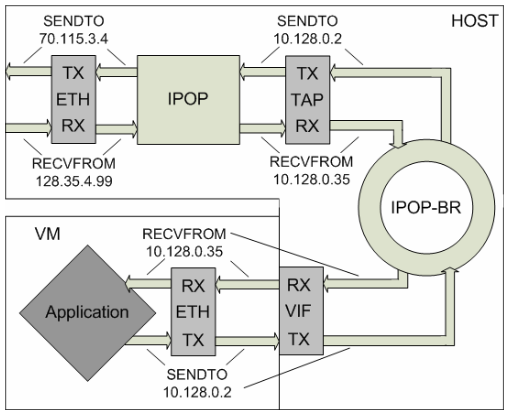

Fig. 2. IPOP deployed in a separate execution domain from the compute VM. The application sends and receives over its logical Ethernet device, which is revealed to the host as a VIF (virtual interface). The IPOP bridge connects the VIF to the TAP device, which IPOP reads and writes. IPOP sends and receives packets for the 10.128.0.0/255.128.0.0 virtual network over the host's physical Ethernet device.

It is also conceivable to isolate the compute VM from the execution environment where network tunneling takes place by deploying nested VMs. For example, Xen dom0 and domU may run inside of a VMware virtual machine. This is a scenario motivated by situations where it may not be possible to deploy additional software on the host (except for the VMM).

B. Virtual network tunneling on guest

An alternative sandboxing deployment consists of running IPOP on the compute VM. This setup is useful to facilitate the deployment of WOW nodes on a wide variety of hosts, by leveraging the ease of deployment of hosted I/O VMMs. In this setup, the sandbox has both network interfaces (the VM's and the IPOP "tap") running in the same operating system. In this case, IPOP is protected from WOW users by means of UNIX protection mechanisms, and virtual network traffic is restricted to the WOW network by means of host firewall rules (e.g. iptables).. This method can be seen in Fig. 3. This approach is easier to deploy than the previous case; however, because IPOP shares the same execution domain as processes initiated by remote users, if a malicious user is able to escalate privileges and gain "root" access on the sandbox, they will be able to lift firewall rules and send packets to hosts on the physical network.

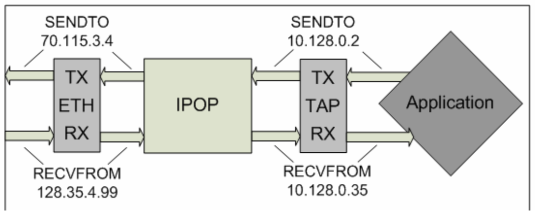

Fig. 3. IPOP deployed in the same domain as the compute VM. The application sends and receives on the TAP device, while IPOP reads and writes to the TAP device. Incoming packets are received by IPOP and written to TAP. Outgoing packets are read by IPOP, converted, and sent over the physical Ethernet device. The IPOP virtual network address space is 10.128.0.0/255.128.0.0 in this example.

C. *IPSec*

With the previously discussed techniques, the ability to use the system for attacks on the internet or other networks outside of the WOW is minimized. In addition, for a secure way of ensuring that only nodes authorized by an administrative entity can connect to a WOW, the sandbox supports IPSec with X.509-based authentication and end-to-end encryption. The deployment of such IPsec-based infrastructure in the current Linux-based is based on existing kernel support for IPsec tunneling and the user-level Racoon [49] application.

This setup illustrates how using a virtualized environment which is homogeneous in terms of both node configurations and virtual IP address space can greatly facilitate the deployment of a complex public key based infrastructure. The sandbox image is customized *once* by an administrator – with a kernel which supports IPsec, with the administrator's certificate authority (CA) public key and with a *single* configuration file for the Racoon application. It is then possible to enforce that all pair-wise communication taking place between any two hosts in the WOW (e.g. a class A, 10.255.255.255 private address space) must be authenticated and encrypted with IPsec in a simple manner, with the use of a netmask. In the absence of a virtual network, and with nodes running different O/S versions and potentially behind NATs, the deployment of the same kind of IPsec infrastructure would be extremely complex to manage.

This deployment model requires a CA, which signs individual nodes' host certificates. To ensure that a user can not use the same signed certificate on multiple nodes, the IP address of the certificate requester is embedded into the certificate request signed by the CA. The mechanism to provide signed IPsec certificates to hosts works as follows. Upon first boot, the WOW node joins the virtual network and obtains an IP address from the DHCP server as previously described. The system may be configured such that at this point the user has limited access to nodes which have been designated as non-IPsec nodes, e.g. so the user can interact with demonstration applications. In order to access private nodes, the user must run a script which will create a certificate request and submit it to the CA node managed by the administrator of the WOW. The request may be automatically or manually signed by the CA and sent back to the requesting VM. Upon reception of the signed certificate, the script places the signed certificate and CA's public key in the appropriate place and starts Racoon. Most importantly, the node is now given access to the private nodes. Updated certificate revocation lists (CRLs) can be distributed across the WOW using similar techniques employed to update the IPOP code as described earlier.

IV. USAGE SCENARIO EXAMPLES

The WOW sandbox, packaged as an easy to distribute VM "appliance", is currently being used by different application domains. A Condor pool with WOW appliance nodes 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 [33] cyber-infrastructure by its community.

WOW sandboxes are also being deployed in support of hurricane storm surge models as part of the SCOOP [34] project. In this context, the WOW appliances 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 WOW nodes is 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 SCOOP's data transport system (UniData's LDM [35]). Event-triggered jobs can be scheduled to run locally on individual WOW nodes, or submitted to other nodes in the WOW through Condor. In the retrospective analysis scenario, data is retrieved from the SCOOP archive, simulation model executions are dispatched to a WOW Condor pool, and results are published back to the SCOOP archive through LDM. Currently, a total of 32 WOW nodes serving these purposes have been deployed at four SCOOP sites.

Another usage scenario where WOW sandboxes 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 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 WOW nodes – 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.

V. EXPERIMENTAL SETUP

To evaluate the performance of the sandbox, three benchmarks were used on the Condor-based deployment described in Section III. The benchmarks are: SimpleScalar [19], a CPU-intensive computer architecture simulator which models the executions of applications with cycle-level accuracy (the experiments employed the SPEC 2000's [20] Go benchmark); PostMark [21], a file system benchmark; and IPERF [22], a TCP throughput benchmark. The performance of these applications is evaluated with 3 different platforms: WOW sandbox 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 WOW sandboxes, with focus on machine configurations that would be expected in a desktop-Grid type environment. The hardware configuration of the physical host is a desktop-class system, with a Pentium IV 1.7GHz CPU with 256KB on-chip cache and 512MB RAM. The software versions used are VMware Server 1.0.1, Xen Testing Nightly Snapshot 09/26/06, Debian Etch for the physical host, and Debian Sarge for the WOW sandbox.

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.

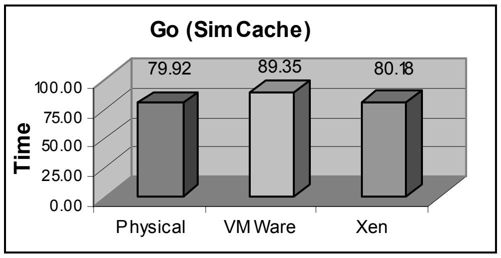

Fig 4. Execution times for Go using SimCache. The application was launched from Condor. The results are in seconds and based upon completion, therefore, less time (smaller) is better.

A. SimpleScalar

SimpleScalar (version 3.0d) is used for benchmarking CPU performance. For the SimpleScalar test, SPEC 2000's 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". The SimpleScalar executable used was Sim-Cache. Figure 4 shows the overall execution times (in minutes) for the execution of SimpleScalar in the three different sandbox configurations considered.

The results are consistent with previous findings; virtual machines show low overhead in the case of processor intensive tasks (0.4% for Xen, 10.6% for VMware).

B. 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.

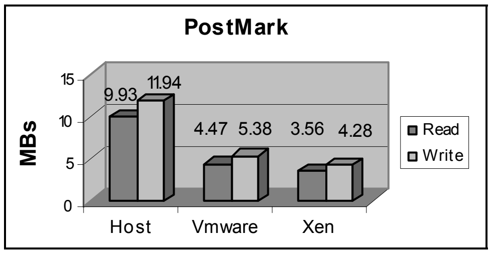

Fig. 5. PostMark I/O throughput results, in read / write MB/s.

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.

C. 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 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[3].

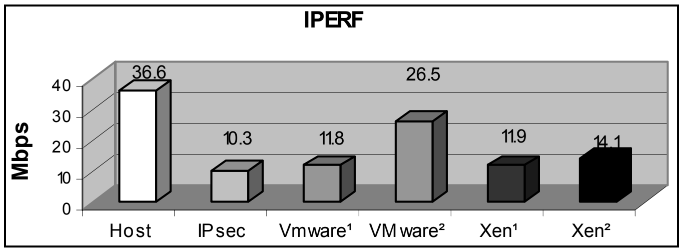

Fig. 6. Results from the IPERF experiment. The results are given in megabits per second, that is bigger is better. Superscript 1 implies virtual networking tunneling inside the VM, whereas 2 means virtual networking tunneled through the host.

[3] 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.

D. *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 [1],[3]. 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.

VI. Related Work

While using virtual machines for distributed computing is not a new idea, the implementation discussed herein offers a unique approach to this problem based on easy to deploy, self-configuring VM "appliances" and the overlay network interconnecting them. It was first discussed in [1] that virtual machines would make for good grid nodes due to their low overhead for CPU-intensive workloads.

In [3], four ways in which Xen can be used for sandboxing in the context of Condor have been evaluated. The approaches mentioned did not build upon the use of a virtual network, and relied on blocking network access with the exception of Condor-mediated remote I/O accesses, to isolate the VM sandbox from the physical network. In contrast, the WOW approach to sandboxing network traffic is not dependent on any specific middleware, and allows bidirectional TCP/IP connectivity of the sandbox with all other VM in the virtual network while preventing connections with machines outside the virtual network.

In [36], the authors discuss the integration of VMs and virtual networks using Virtual Distributed Ethernet [37]. Both [39] and [40] have proposed a Xen based single-computer Grid gateway. These approaches focus on LAN environments and do not address cross-domain issues such as NAT/firewall traversal which are a key enabling technology in WOW sandboxed.

In [41] and [42], Keahey et al describe virtual workspaces as a configurable and controllable remote execution environment based on virtual machines based on Globus [48]. Similarly, OurGrid [43, 44] uses the Xen based SWAN (Sandboxing Without A Name) in their implementation; however, their main motivation towards this is to address the security issues rather than a generic appliance based solution for wide-area and large scale deployments. In [45] the authors propose an adaptive, self-configuring and self-distributing virtual machine for clusters of heterogeneous, dynamic computer resources; however their approach is not focused on system VMs.

The Minimum Intrusion Grid [38] project focuses on developing grid middleware allowing users and resources to install and maintain a minimum amount of software to join the grid. The basic theme is similar to the WOW sandbox, but their implementation is not based on system VMs connected by virtual networks.

VII. Conclusions and Future Work

The use of WOW sandboxes in distributed systems has great potential because of the simplicity in which nodes across wide area domains can be added to the virtual network. This paper shows that different degrees of sandboxing can be provided, and quantifies the performance overheads with current state of the art VM monitors. The focus of WOW sandboxes is on providing a new capability that facilitates the deployment of wide-area collaborative environments; one particular application of the sandbox for Condor based wide-area high-throughput computing has been successfully demonstrated. Experiments show that the overhead in CPU-intensive applications is small, and quantify the disk I/O and virtual networking overheads of WOW sandboxes. As improvements in VMM implementations and virtualization-enabled hardware yield more efficient virtualized systems, it is anticipated that the VM results will more closely match that of the host.

In the course of several months, the WOW sandbox for Condor-based computing has been made available to the public for download. Dozens of individual nodes have logged in joining a demonstration virtual network (and submitting example Condor jobs) from all over the world, which indicates that this self-configuring system is a simple one to deploy. Future plans include conducting user surveys and careful monitoring of WOW nodes to better understand not only the performance of the distributed system, but also how users perceive its capabilities and limitations.

Acknowledgments

Effort sponsored by the NSF under grants EIA-0224442, EEC-0228390, ACI-0219925, ANI-0301108, SCI-0438246 and SCI-0537455 and carried out as a component of the "SURA Coastal Ocean Observing and Prediction (SCOOP) Program", an initiative of the Southeastern Universities Research Association (SURA). Funding support for SCOOP has been provided by the Office of Naval Research, Award #N00014-04-1-0721 and by the NOAA Ocean Service, Award # NA04NOS4730254. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

REFERENCES

[1] R. Figueiredo, P. A. Dinda, J. A. B. Fortes. "A Case for Grid Computing on Virtual Machines". Proc. International Conference on Distributed Computing Systems (ICDCS), May 2003.
[2] A. Ganguly, A. Agrawal, P.O. Boykin, R. Figueiredo. "IP over P2P: Enabling Self-Configuring Virtual IP Networks for Grid Computing". Proc. International Parallel and Distributed Processing Symposium, 2006.
[3] S. Santhanam, P. Elango, A. Arpaci-Dusseau, M. Livny. "Deploying Virtual Machines as Sandboxes for the Grid". Proceedings of USENIX Worlds, 2005.
[4] P. Barham et. al, "Xen and the Art of Virtualization". In Proceedings of the ACM Symposium on Operating Systems Principles, October 2003.
[5] A. Ganguly, A. Agrawal, P.O. Boykin, R. Figueiredo. "WOW: Self-Organizing Wide Area Overlay Networks of Virtual Workstations". In Proc. of the 15th IEEE Int. Symp. High Performance Distributed Computing, 2006.
[6] S. Adabala et al, "From Virtualized Resources to Virtual Computing Grids: The In-VIGO System". In Future Generation Computer Systems, vol 21, no. 6, April, 2005.
[7] D. Joseph, J. Kannan, A. Kubota, K. Lakshminarayanan, I. Stoica, K. Wehrle. "OCALA: An Architecture for Supporting Legacy Applications over Overlays". USENIX NSDI, 2006.
[8] K. Keahey, K. Doering, I. Foster. "From Sandbox to Playground: Dynamic Virtual Environments in the Grid". 5th International Workshop in Grid Computing. November 2004.
[9] C. Wright. Unionfs: Bringing Filesystems Together. http://www.linuxjournal.com/article/7714, November 2004.
[10] M. Krasnyansky. Universal TUN/TAP Device Driver. http://www.kernel.org/pub/linux/kernel/people/marcelo/linux-2.4/Documentation/networking/tuntap.txt
[11] P. O. Boykin, J. Bridgewater, J. Kong, K. Lozev, B. Rezaei, V. Roychowdhury. Brunet software library. http://brunet.ee.ucla.edu/brunet/
[12] VMware. http://www.vmware.com.
[13] Condor Team, University of Wisconsin- Madison. Condor Version 6.8.0 Manual. http://www.cs.wisc.edu/condor/manual/v6.8/, 2006.
[14] R. Droms. RFC 2131 Dynamic Host Configuration Protocol. Mar 1997.
[15] S. Alexander, R. Droms. RFC 2132 DHCP Options and BOOTP Vendor Extensions. March 1997.
[16] P.V. Mockapetris. RFC 1034 Domain names - concepts and facilities. November 1987
[17] P.V. Mockapetris. RFC 1035 Domain names - implementation and specification. November 1987.
[18] David 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.
[19] SimpleScalar. http://www.simplescalar.com/
[20] SPEC 2000. http://www.spec.org/cpu/.
[21] J. Katcher. "PostMark: a new file system benchmark", TR3022. Network Appliance, October 1997
[22] IPERF. http://dast.nlanr.net/Projects/Iperf/
[23] S. Kagstrom. XenSegments. http://wiki.xensource.com/xenwiki/XenSegments.
[24] Python. http://www.python.org
[25] C. Kaufman. RFC 4306 Internet Key Exchange (IKEv2) Protocol. December 2005.
[26] M. Litzkow, M. Livny, and M. Mutka. Condor - a hunter of idle workstations. In Proc. 8th IEEE Intl. Conference on Distributed Computing Systems (ICDCS), Jun 1988.
[27] X. Jiang and D. Xu. Violin: Virtual internetworking on overlay infrastructure. In Proc. of the 2nd Intl. Symp. on Parallel and Distributed Processing and Applications, Dec 2004.
[28] A. Sundararaj and P. Dinda. Towards virtual networks for virtual machine grid computing. In Phroc. of the 3rd USENIX Virtual Machine Research and Technology Symp., San Jose, CA, May 2004.
[29] M. Tsugawa and J. A. B. Fortes. A virtual network (ViNe) architecture for grid computing. In To appear, Proc. of the IEEE Intl. Parallel and Distributed Processing Symp. (IPDPS), Rhodes, Greece, Jun 2006.
[30] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. Planetlab: An overlay testbed for broad-coverage services. ACM SIGCOMM Computer Communication Review, 33(3), 2003.
[31] Open science grid web site. http://www.opensciencegrid.org.
[32] C. Sapuntzakis, D. Brumley, R. Chandra, N. Zeldovich, J. Chow, M. Lam, M. Rosenblum. Virtual Appliances for Deploying and Maintaining Software. 2003
[33] J. Fortes, R. Figueiredo, M. Lundstrom 'Virtual Computing Infrastructures for Nanoelectronics Simulation'. In Proceedings of the IEEE 93(10), 08/2005, p1839-1847
[34] P. Bogden, et al, "The Southeastern University Research Association Coastal Ocean Observing and Prediction Program: Integrating Marine Science and Information Technology,". Proceedings of the OCEANS 2005 MTS/IEEE Conference. Sept 18-23, 2005, Washington, D.C.
[35] G. P. Davis and R. K. Rew, "The Unidata LDM: Programs and Protocols for Flexible Processing of Data Products". Preprints, Tenth International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology, pp. 131-136, American Meteorological Society, January, Nashville, Tennessee.
[36] R. Davoli and M. Goldweber, " Virtual Square (V^2) in computer science education", In Proceedings of the 10th Annual SIGCSE conference on Innovation and Technology in computer science education, June 2005, Costa da Caparica, Portugal
[37] R. Davoli, VDE Sourceforge Home Page http://vde.sourceforge.net
[38] R. Andersen and B. Vinter, "Minimum Intrusion Grid- The Simple Model", In Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE – 2005), June 13-15 2005, Sweden
[39] S. Childs, B. Coghlan, D. O'Callagham, J. Walsh and J. Quigley, "A single-computer grid gateway using virtual machines", In proceedings of the IEEE 19th International Conference on Advanced Information networking and Applications, March 28-30 2005, Taiwan
[40] S. Childs, B.A. Coghlan, D.O. Callaghan, G. Quigley and J. Walsh, " Deployment of Grid gateways using virtual machines", In Proceedings of EGC'05, Amsterdam, February 2005
[41] K. Keahey, I. Foster, T. Freeman and X. Zhang, "Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid", Scientific Programming Journal. Vol. 13, No. 4, 2005. Special Issue: Dynamic Grids and Worldwide Computing, pp 265-276.
[42] K. Keahey, I. Foster, T. Freeman, X, Zhang, D. Galron, "Virtual Workspaces in the Grid", Europar 2005, Lisbon., Portugal, September 2005.
[43] W. Cirne et. al, "Labs of the World, Unite!!!", Journal of Grid Computing, June 2006-08-25
[44] M. Mowbray, "OurGrid: A Web-based Community Grid", Proc. IADIS International Conference on Web Based Communities (WBC 2006)
[45] J. Haase, F. Eschmann, K. Waldschmidt, "The Self Distributing Virtual Machine (SDVM) – Making Computer Clusters Heal Themselves", In Proc. 23rd IASTED International Multi-Conference on Parallel and Distributed Computing and Networks, February 15-17, 2005, Innsbruck, Austria
[46] Globus, I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. Intl. Journal of Supercomputer Applications, 11(2):115–128, 1997.
[47] Racoon, http://ipsec-tools.sourceforge.net/