Simplifying Resource Sharing in Voluntary Grid Computing with the Grid Appliance

David Isaac Wolinsky and Renato J. Figueiredo
Advanced Computing and Information Systems Laboratory
University of Florida
davidiw@ufl.edu, renato@acis.ufl.edu

***Abstract*—Research projects in many fields are increasingly reliant on the use of computer-based simulation and computing grids. Many projects have successfully leveraged voluntary computing infrastructures by developing and distributing "@home" applications using the BOINC framework. Through generous contributions from the general public, these systems now have a computing backbone on which to have their data processed or simulations run. A shortcoming of such systems is that most users are often limited to contributing resources and few users are capable of developing or porting their own applications in order to use these resources. While many users are satisfied with receiving points (an intangible good) in return for their contribution, the need to port applications presents a barrier to entry to many other users who can potentially benefit from using the voluntary resources.**

In this paper, we describe enhancements made to the "Grid Appliance", a virtual machine based system which enables an execution environment in which users are given the opportunity to voluntarily share (providing and using) resources and run unmodified x86/Linux applications. Voluntary grids introduce a host of issues to tackle, most importantly getting users involved quickly. With that in mind, the Grid Appliance provides many tools for making a user-friendly environment for users, developers, and administrators. This paper summarizes the challenges of getting users involved, reducing the overhead for administrators, and describes the solutions used in the Grid Appliance.

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

I. Introduction

While virtualization technologies can substantially ease deployment and use of distributed systems, experience suggests that many potential users of these systems are unaware of the capabilities and limitations of a shared grid environment. The Grid Appliance [1] virtual environment has been developed with the key goal of reducing the entry barrier of complex grid environments, such that end users, even inexperienced ones, can setup and use a Condor-based Grid node within a matter of minutes. As such, the architecture and interfaces of the Grid appliance have been developed with user-friendliness as the focal point, both to end users and to administrators of an appliance-based infrastructure. This paper summarizes our experiences in developing and deploying wide-area Grid Appliance pools for over one year, and its contributions are in the discussions of lessons learned, design decisions, integration and analysis of interfaces and self-configuring software for virtual appliance-based desktop and voluntary grids.

To assist in decreasing the complexity of the Grid Appliance interface, demonstrations and courses were taught, surveys were taken, and tutorials were prepared. Through this process, responses from users with varying backgrounds provided insights not apparent to the development team. The demonstrations and courses provided an opportunity to interact directly with target audiences. It was clear that users wanted simple point-and-click interfaces.

Over time it became apparent that there were shortcomings in the original design, such as the requirement of being able to handle a POSIX driven command-line. While most engineers were comfortable with this, the target audience includes individuals from all walks of life, including biologists, chemists, and students, many of whom live in a completely "Windows" world. In order to easily distribute our system, the open-source Linux O/S was used which, while able to provide similar look-and-feel to Windows, is not identical. Adding greater GUI functionality increases the download size and the performance overhead in the appliance, which is a deterrent in getting users interested and involved.

In the initial Grid appliance design, it was cumbersome to deploy new pools of resources, which limited the possibility of providing opportunistic private computing desktop Grids within an organization. With a focus on decentralization and self-configuration, the process of bringing up separate pools has been streamlined such that it does not require users to set up any external infrastructure, just a single configuration file in the Grid Appliance. This along with several other features described in the paper has reduced the overhead for system deployment and management.

The rest of the paper discusses detail the solutions to the aforementioned issues, how to deal with enabling multiple users to share a single instance of the Grid Appliance, simplifying the deployment of a security infrastructure, and enabling the Grid Appliance to be seamlessly portable across various virtualization platforms, including the open source KVM and Virtualbox.

II. What Is the Grid Appliance

This section provides a high level overview of the Grid Appliance system architecture initially described in [1], providing context for the discussions to follow. The three major components of the Grid Appliance are virtualization, services, and user interfaces.

A. Virtualization and What It Can Do For You

Virtual machines provide an abstracted environments which, from the user's point of view, appears as if operating systems are running as applications in their host operating system. From the host operating system, a virtual machine is nothing more than another application and has no more privileges than any other application. The virtual machine "guest" operating system views itself as a completely isolated, sandboxed system running on real hardware where unmodified software (including the O/S kernel) is supported.

Virtual machines are becoming a popular way to distribute complete software packages in the form of appliances [2]. For example, consider the practice of setting up a Web server. The basic requirements are the ability to install services such as the Web server, an SQL database, and scripting language support, and possibly a content management system. A virtual machine allows one user to prepare the system and share as packaged, ready to run system with others. This is because it stores all its data to disk as standard portable files. So the requirements in setting up the entire Web server environment are reduced to downloading the appliance and a virtual machine monitor, which are easily installed. Surveys for the Grid Appliance suggest that, including download and installation time, the process of accessing the grid and running a demonstration job can be done in less than 30 minutes.

B. Making Services User-Friendly

Building on existing software, the Grid Appliance can take advantage of many different services. This section focuses on networking, file systems, and security, and the zero-configuration features in the appliance which make these services seamless to the user.

1) Networking: Distributed peer-to-peer systems (P2P) rely on direct connectivity between all participants. Connectivity is simple when all nodes in a system can talk directly to each other, such as in a LAN; however, with the modern Internet, creating a system with direct connectivity regardless of location is impossible. The lack of IPv4 addresses as well as security concerns have caused users to adopt the sharing of a single address with devices such as network address translators (NAT), which are an impediment to all-to-all connectivity in P2P systems. To work around this problem, the Grid Appliance takes advantage of IPOP [3], [4]. IPOP provides a NAT traversing self-organizing overlay that routes IP-over-P2P. This allows two different Grid Appliance nodes placed behind two completely different networks both connected to the Internet or a common network to communicate with each other over a self-configuring virtual private network.

The interface of IPOP to the user is transparent. When a user boots the Grid Appliance, system initialization calls a script that starts IPOP. This process includes enabling a TAP [5] device and starting the IPOP process. The system recognizes the TAP device as any other network device and uses the DHCP [6], [7] protocol on the device to configure the IPOP virtual IP address.

Another key feature required for many legacy applications is the use of domain name services DNS [8], [9]. These services do not require user-friendly domain names, so the Grid Appliance uses a virtual DNS server, which provides a direct mapping between the IPOP IP address and a host name, enabling fast name resolution with a loop-back server.

2) File Systems Extensibility through Modules: Appliances provide a convenient means of distributing software. The problem, however, is that once an appliance is made, it can be difficult to create a derivative product and still maintain updates from the original appliance. For example, a user takes a Web server appliance to be their home Web server. In the event that the Web server appliance has upgrades for performance or security, it becomes a difficult procedure for the user to migrate his Web site over to the new appliance. That would require at least a knowledge of how file systems and databases work, if not more, and defeats the purpose of having appliances in the first place.

In the Grid Appliance, this problem is addressed through the use of union file systems [10], [11] and modules. Union file systems work by merging the contents of several file systems. In the Grid Appliance, three layers are used: the first layer provides the data for the base system, the second layer allows for developers to add/customize new features to the Grid Appliance independently from any specific version of the base system, in what is called a module. The final layer provides an area for the user's content, so that when a new base or module is released, their files are easily migrated.

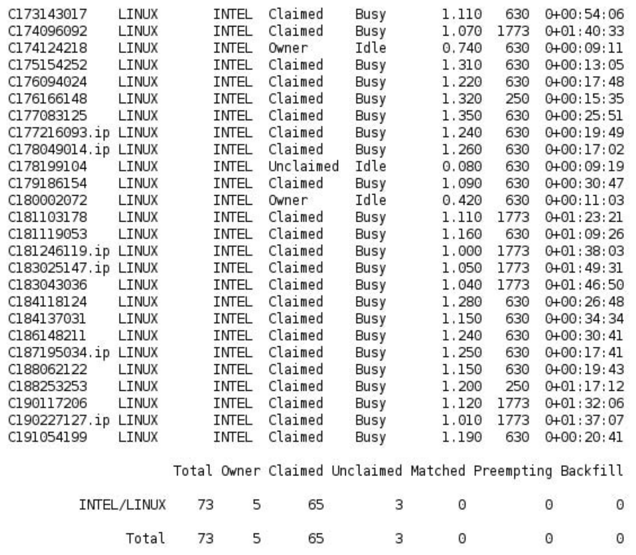
3) Security: Having a P2P system in one's hands can be an undertaking, because as easy as it is to harness the power of the system for good, it can equally be used for evil. If a malicious user were to gain access to several naively setup machines, they would have the ability to use it to cause problems for other users of the Internet, e.g. with a distributed denial of service (DDoS) attack. This would be even easier if all the machines were on the same local network and had the exact configuration, which just happens to be the case with the Grid Appliance.

A simple approach to prevent such kinds of attacks actively deployed in the Grid Appliance is a strict firewall implemented by Linux's IPTables. This prevents all communication by regular users to public Internet servers outside the IPOP network. The virtual networking component and other necessary components are run by the administrative user who is capable of communicating over the network. Of course, this approach cannot prevent users who escalate privileges within the appliance to lift the firewall constraints; we rely on security mechanisms in the Grid middleware and the underlying O/S (Condor and Linux in our case) to prevent escalation of privileges.

The approach to providing strong authentication and end-to-end encryption is based on signed certificates in the form of IPSec[12] and its helper utility Racoon. A safe, certificate authority machine is assumed; it has the proper authority to sign X.509 host certificates. Users who wish to join a Grid appliance pool send a requesting certificate to the central machine, who can determine based upon the user's credentials whether to sign or to deny the request. In the case of signing, the user would simply place the signed certificate in their machine and have protected safe access to the pool. Now in the case of malicious usage, there is a direct correlation between the IPOP address of the malicious appliance and their owner's identity, and allowing the host certificate to be revoked.

C. *Making Grid Systems User-Friendly*

In an attempt to get users connected smoothly to the grid, when a user turns on a Grid Appliance, the first thing a user sees is a graphical interface providing some hints as well as a console waiting for batch job submission. This also meant that a user-friendly grid environment with support for multiple entry points either needed to be found or built. Thankfully, Condor covered this base. Other popular grid environments like PBS [13], [14], [15] and Grid Engine [16] rely too much on a shared file system and the use of a single user submission site, while Condor enables grids to be configured without the need of a shared file system and allows any participant in the pool to also submit tasks. In appliance pools, a few nodes in the system run the Condor resource database and match-making daemons (collector, negotiator), while end-user appliances run the scheduler and starter daemons, allowing end-user nodes in the system to both submit and receive jobs. Condor also sports features for scalability and fault-tolerance, such as flocking to allow multiple collector / negotiator nodes, priority schemes and preemption fair usage of resources, and job migration and checkpointing, which are leveraged unmodified by the appliance.



```
C173143017    LINUX         INTEL  Claimed    Busy        1.110   630  0+00:54:06
C174096092    LINUX         INTEL  Claimed    Busy        1.070  1773  0+01:40:33
C174124218    LINUX         INTEL  Owner      Idle        0.740   630  0+00:09:11
C175154252    LINUX         INTEL  Claimed    Busy        1.310   630  0+00:13:05
C176094024    LINUX         INTEL  Claimed    Busy        1.220   630  0+00:17:48
C176166148    LINUX         INTEL  Claimed    Busy        1.320   250  0+00:15:35
C177083125    LINUX         INTEL  Claimed    Busy        1.350   630  0+00:25:51
C177216093.ip LINUX         INTEL  Claimed    Busy        1.240   630  0+00:19:49
C178049014.ip LINUX         INTEL  Claimed    Busy        1.260   630  0+00:17:02
C178199104    LINUX         INTEL  Unclaimed  Idle        0.080   630  0+00:09:19
C179186154    LINUX         INTEL  Claimed    Busy        1.090   630  0+00:30:47
C180002072    LINUX         INTEL  Owner      Idle        0.420   630  0+00:11:03
C181103178    LINUX         INTEL  Claimed    Busy        1.110  1773  0+01:23:21
C181119053    LINUX         INTEL  Claimed    Busy        1.160   630  0+01:09:26
C181246119.ip LINUX         INTEL  Claimed    Busy        1.000  1773  0+01:38:03
C183025147.ip LINUX         INTEL  Claimed    Busy        1.050  1773  0+01:49:31
C183043036    LINUX         INTEL  Claimed    Busy        1.040  1773  0+01:46:50
C184118124    LINUX         INTEL  Claimed    Busy        1.280   630  0+00:26:48
C184137031    LINUX         INTEL  Claimed    Busy        1.150   630  0+00:34:34
C186148211    LINUX         INTEL  Claimed    Busy        1.240   630  0+00:30:41
C187195034.ip LINUX         INTEL  Claimed    Busy        1.250   630  0+00:17:41
C188062122    LINUX         INTEL  Claimed    Busy        1.150   630  0+00:19:43
C188253253    LINUX         INTEL  Claimed    Busy        1.200   250  0+01:17:12
C190117206    LINUX         INTEL  Claimed    Busy        1.120  1773  0+01:32:06
C190227127.ip LINUX         INTEL  Claimed    Busy        1.010  1773  0+01:37:07
C191054199    LINUX         INTEL  Claimed    Busy        1.190   630  0+00:20:41

                     Total Owner Claimed Unclaimed Matched Preempting Backfill

        INTEL/LINUX    73     5      65         3       0          0        0

              Total    73     5      65         3       0          0        0
```

Fig. 1. The execution of the command condor_status. It shows the state of the nodes in the system.

III. Getting Users Involved: Documentation and Interfaces

Users approach interacting with new grid applications in many different ways. Some users will attempt to jump straight into everything without a second thought, while others may want some quick pointers. Few want a long document explaining how everything works in detail. This poses problems for systems developers, where there can be many components with varying amounts of relevance amongst them. How does one properly make documentation that is readily available to all these groups? How do you avoid the questions that are obvious, if the user were to only read the "How to?"

A. *Pointing Users in the Right Direction*

Because the Grid Appliance has a conglomeration of pre-established components, it is able to reuse already written documentation. However, one only needs to look at the complex documentation of a Grid scheduler [17] to realize that simply pointing a user there is not enough. Several approaches have been taken to deal with this: tutorials, well organized "Readme" documents for different topics, quick start guides, publications explaining critical components, and links to associated projects. A Web portal provides all this information and also serves as a front-end to a pool of resources and itself is packaged as appliance for easy distribution.

The tutorial is a quick run through of tools embedded into the Grid Appliance pertaining only to obtaining access to grid features. The tutorial links directly to a survey, where the most valuable feedback from external resources has been compiled. The results were compiled from a variety of backgrounds, for example, 60% of the surveyors had no experience with grid computing. This provided an excellent source of feedback as users rated the Grid Appliance with over 90% of good or better ratings for their experience in using the Grid Appliance and its effectiveness. Everyone agreed that it is an excellent tool for batch computing. While a majority did agree that it was easier than most of their applications, there were still some that saw it was more complex. It is that group which is the primary focus in making the system more user-friendly. The primary issue lied in the fact that not all users have the proper background to understand how grid computing can benefit them and that the Grid Appliances provides an easy way to take advantage of it.

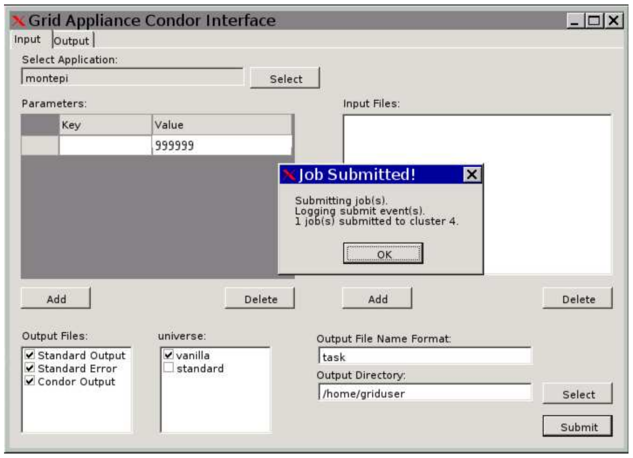
The Web interface provides several other tutorials for all types of users. Specifically, the system has 3 levels of users: casual/entry-level users who would want to test the system out and run tasks from their own desktop; advanced users who would want to set up their own independent pools while reusing the shared PlanetLab infrastructure; and expert users who would want to deploy independent pools on a completely private system. As users become more comfortable, more advanced tasks and tutorials are available to guide them from entry-level to expert usage.

B. The Graphical User Interface Effect

If users were confronted with the decision between a graphical or console application providing the same capabilities, no doubt that a majority would choose the graphical version. When the first paper entailing the Grid Appliance was released over a year ago, so much of the time had been spent in developing and stabilizing a well-rounded grid system and little development towards developing a user-friendly interface.

We have since developed a generic user front end for Condor to serve as a primary GUI for job submission (see 2 for a screenshot). By using the .NET libraries and C#, this application can be run on any computer that supports .NET, since it is dependent on having a run-time environment on the host machine to support conversion from byte-code to native code. With a free, stable implementation available (Mono), this front end is extremely portable. It currently supports the core functionality of selection of an executable, input files, command-line arguments, job submission and monitoring.

It is trivial to run this interface within the appliance. Ideally, however, accessing the pool through the GUI would be easier if it were accessible from the host operating system and not just within the appliance. Through the use of a zero-configuration start up sequence, the GUI software locates an instance of a Grid Appliance. From there, communication occurs by leveraging Condor's BirdBath API library for SOAP interactions. As a result, the user can submit jobs and receive data back from the jobs through this API using their host windowing system, without interacting with the appliance's X11 window system which may be foreign to them.



Select Application: montepi

Parameters:

Key	Value
	999999

```
Submitting job(s).
Logging submit event(s).
1 job(s) submitted to cluster 4.
```

Output Files: ✓ Standard Output, ✓ Standard Error, ✓ Condor Output

universe: ✓ vanilla, ☐ standard

Output File Name Format: task

Output Directory: /home/griduser

Fig. 2. The graphical user interface running the Monte Carlo Pi estimator demonstration program.

IV. Easing the Burden on Administrators: Management enhancements

A reason why many pre-deployed services like TeraGrid [18], OpenDHT [19], NanoHub [20], InVIGO [21] do not have distributable installation packages lies in the fact that setting up such services is difficult. Providing front-ends and clients is significantly easier. Over the past year, much effort has been put into reducing the difficulty of installation, assistance in monitoring the system, and reducing the requirements on any one specific machine. The purpose being that deploying independent Grid Appliance systems should be simple and straightforward. These are described in this section.

A. Easing Deployment

In the initial appliance design, users could easily join a pre-deployed pool but it was far from trivial to setup one's own pool. This limitation prevents many would-be users or administrators from hosting their own appliance private pools where they may have no desire to share their resources with unknown parties. To address this, the architecture of the core IPOP virtual network has been enhanced to support multiple IP namespaces [22] and the self-configuring software in the appliance has been enhanced to enable easy creation of independent pools.

The base appliance distribution for job submission / execution consists of a single VM image. We also have deployed a "bootstrap" infrastructure on hundreds of nodes in the PlanetLab [23] test bed such that users can easily connect to an existing environment. Depending on the route the user wants to go, they can either use this open infrastructure or create their own. The default configuration connects to the open infrastructure; if a user wishes to create their own pool, they can generate appropriate configuration files on virtual floppy images via the Web interface. The system provides support for three different appliance configurations: Condor client (for interactive job submission/execution), Condor manager (for job scheduling), and Condor worker (for unmanaged job execution). Users can then copy and attach the appropriate floppy disks to the baseline image to build an independent pool.

For instance, suppose our open infrastructure runs on PlanetLab hosts and connects appliances bound to the default IPOP namespace named "UFL_Condor". An administrator wanting to create a completely independent PC lab grid on their institution would proceed as follows:

1) Choose a unique name for the IPOP namespace (e.g. "UniversityXpool")
2) Register with the Web front-end to create the virtual floppy disks
3) Download, copy and deploy the grid appliance base image and the "manager" virtual floppy on one or more servers
4) Deploy the appliance on lab PCs using the "worker" virtual floppy
5) Distribute the "client" floppy packaged with the base appliance image to the end users of the system

More information is provided on the Web interface by means of tutorials.

B. Monitoring the System

Condor already provides a nice suite of applications to monitor the state of its system; however, with the appliance it becomes necessary to also monitor the underlying IPOP infrastructure. The discussion in this section focuses on enhancements meant to assist developers of infrastructure code and users / administrators who have setup setup their own backends.

The Grid Appliance creation image also provides a tool called a crawler, which walks the entire P2P overlay virtual network. This helps determine the state of the system and assists in finding possibly bad nodes. Crawling helps determine the consistency of the IPOP overlay. In IPOP, any node and its neighbor must agree on their position in its ring topology; that is, the X node must agree that X+1 is its left neighbor and X+1 must agree that X is its right neighbor. Nodes lacking consistency could be caused by being a faulty node, or a neighbor of a faulty node. Finding these poorly reacting nodes is important, because at this point in time, there is no mechanism to autonomously remove them from the system. If they remain, packets can be lost and connectivity broken.

The other application in development assists administrator in determining the network location of all Grid Appliances in the system. If a user were to install 100 nodes in a cluster, finding the one that may have crashed or corrupted data can be an extremely time consuming matter, requiring an administrator to log into every virtual machine monitor until the broken one was found. With this tool the developer would be able to locate all the machines which have Grid Appliances and how many instances are run on that machine. Taking that list and a list of all locations where the Grid Appliance should be installed, an administrator would be able to pin point the broken node.

C. *Reducing the Stranglehold of Centralization*

Unlike systems that require users to access a head node, such as PBS and GridEngine, Condor out of the box provides the ability to use any node in the system submit tasks. The only requirement being that all nodes have direct connectivity; in the Grid Appliance this is dealt with by the virtual networking of IPOP. However, in the initial Grid appliance design, the IPOP DHCP server was centralized and there was no self-configuring mechanism to support the Condor flocking feature. These limitations meant that the IPOP DHCP server and Condor manager nodes needed to be closely monitored by an administrator, since they become single points of failure.

Previous work [22] discusses the integration of a decentralized object store/retrieval system (Distributed Hash Table, or DHT) into IPOP with example usage of DHCP. This technology was integrated into the Grid Appliance making the Condor manager the only remaining centralized portion of the Grid Appliance.

The approach of fully decentralizing Condor requires substantial modifications to Condor itself. Condor does however support the use of flocking, which provides the ability of a system to continue servicing new jobs if a manager fails. We have built on this ability and leveraged the DHT data structure to implement publish and discovery of Condor primary and flocking managers. An appliance booting with the "manager" virtual floppy claims to be a manager by putting such information and its IP address into the DHT. Nodes looking for a manager (i.e. "client" and "worker" nodes) can look up the generic manager key and find any available managers. If there are multiple managers in the pool, a node will choose at random one of them as its main manager and the rest as flock-to nodes. In the case the main manager goes down, the node will restart Condor with one of the remaining managers. Contrasting this work with previous work [24], the differences are that in our design the DHT is used to self-organize the connection of every node to not only flocking node(s), but also to its primary manager. Ongoing work is being done to provide a more flexible design that would require no specific client or manager nodes and no restarting of Condor.

V. Tie-ins and Open Issues

This section reviews components that deal with issues that affect the overall system including administrators and users.

A. *Bandwidth Hog and Dealing with Network Limitations*

In the absence of a common file system, when a user submits the same application multiple times with different parameters, Condor uploads that same application for each task to the machine where the task will run. For home users this can quickly make their connectivity to the Internet completely useless due to the overwhelming strain on upstream bandwidth. The ripple down effects are that the time saved by running the tasks in parallel may be lost due to the bottleneck provided by the transfer of the application to the remote computer. There are two partial solutions to this, one being have bandwidth aware software and, the second, based on a distributed file system.

There are many ways to implement bandwidth throttling. A simple solution is to take advantage of built-in tools provided by Linux, e.g. tc, for traffic control. This works well but is not adaptive, so users would be forced in selecting an optimal speed for them. Future thoughts on the matter would incorporate the techniques done in tc into IPOP and then add algorithms to determine the optimal upstream bandwidth that would not interfere with the users connectivity. This does not deal with the problem of multiple file transfers.

There has been much work in distributed file systems, but the unique environment in which the Grid appliance operates is unlike typical environments where such file systems have been deployed. P2P file systems such as Shark[25] which provide an NFS interface and cooperative caching are the closest to our environment. However, because of our goal of deploying systems to large user bases, it is very important to deploy production-quality software; at this point in time, this is still an open issue. On going research is taking place in the form of a DHT file system to be attached to the existing IPOP DHT as well as the possibility of a BitTorrent based file system. The main advantages of the BitTorrent file system is the cooperative caching effects that would not readily available in a DHT file system.

B. Removing the Shackles of Proprietary Software

In the very beginning, this project aimed to be a contender in the VMWare Virtual Machine Appliance challenge, where it was one of the top 15 appliances out of over 100 submitted. Life began for the Grid Appliance purely focused to be run on VMWare [26] virtual machines. Over the past year and a half, the market of virtual machines monitors has become increasingly diversified and now there are contenders such as VirtualBox [27], Parallels [28], Xen [29], and KVM [30]. Each one of these virtual machine monitors have their advantages: VirtualBox being free and open source, Parallels being the first to have good support for Mac OS/X, Xen has low performance overheads and is open source, and KVM is the lightest weight as it comes with modern Linux distributions.

The reliance on VMWare also caused issues with the nature of the Grid Appliance, which is that it is free open source software, while VMWare is not. This led to many purists ignoring the potential of the Grid Appliance. To remedy this problem, the Grid Appliance imagine has been designed such that it becomes straightforward to automatically convert from a baseline .vmdk image to formats used by KVM, Parallels and VirtualBox. The virtual disk has been configured to be loaded as SCSI disks supported by VMWare or IDE disks supported by the other VMs. To ensure compatibility, the Grid Appliance has successfully been run on all the previously mentioned virtual machine monitors mentioned.

VI. Bringing It All Together - The Web Interface

This paper introduces many new services and discusses new levels of decentralization that should decrease setup time and increase productivity. However, there are some components that our experience shows a centralized approach to be best, namely a system management "console" and a Web-accessible user interface and content management system portal based on Apache and Joomla.

The purpose of the web portal is to bring a centralized location where users can retrieve packages, share a single resource, and receive news and administrators can monitor the system and take care of security responsibilities. Our envisioned deployments are ones where a (virtual) organization can easily deploy a pre-packaged management and portal front-end by downloading the base appliance and front-end module, use the management interface to create manager / worker / client virtual floppies, and seamlessly deploy appliance pools.

The Web interface is available through a module add-in for the Grid Appliance, building on the UnionFS capability described earlier in this paper. Further, the Grid Appliance system can run completely independent or with multiple web interfaces accessing it. To use it, a user needs to only download the module, add it to the Grid Appliance, and access the IP address for the virtual machine in a web browser.

In the current deployment, we host a Grid appliance web interface is available for public use at http://www.grid-appliance.org. Users are able to download appliance images, documentation, register with the system to generate and download floppy images with custom configurations to create their own independent pools, and check the statistics of the system.

A. User Interfaces

In addition to providing documentation and additional content on how to use the system, to further promote better user interfaces in a collective environment, the Grid Appliance borrowed ideas from environments such as InVIGO and NanoHub. The Grid appliance Web interface allows remote users to submit, monitor and interact with applications, and to upload and download files. In essence, it provides the core capabilities to deploy a gateway to a grid appliance pool without requiring one to run an appliance on their desktop. Such interface is very important in attracting new users by displaying the capabilities of the system with low overhead - all the user needs to interact with the system is a Web browser. Two approaches to this are individual user web space and interfaces. There are two interfaces provided, one that hooks directly over VNC [31] and AJAX [32].

VNC works by forwarding a host's graphical session to a remote machine. By using a Java library, it is possible to integrate this right into web pages. This means that graphical applications can be run completely unchanged. Results are stored in the WEBDAV [33] file system, a web based file system that is easily mountable in Windows, Linux, and Mac OS/X. The disadvantage is that each instance requires at least 10MB of space and everything is run on the server in volatile memory and can be draining on the processor of the host machine.

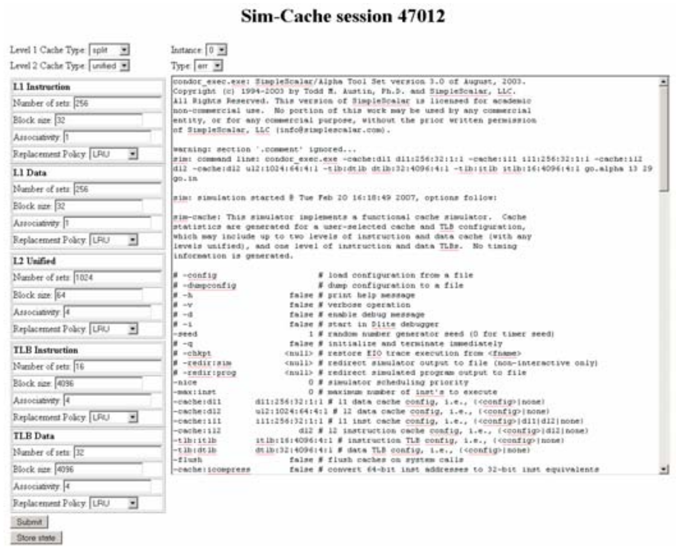
Web 2.0 enables interactive Web sessions that are substantially richer and more appealing to users than static HTML. As a proof-of-concept, the appliance portal has support for AJAX-based applications; currently, a demonstration application using the SimpleScalar computer architecture simulator is in place. Future work is still needed to devise both a generic framework for developers to use a single XML file to define their interface and its tie into to Condor. A key feature in this system is that all the session data is stored in the database and even if the server crashes the data will be available the next time the user returns.

B. System Management

The tool for "crawling" the virtual network for monitoring the state of the system is also integrated with the interface/management appliance. This tool takes a snapshot of the entire system every 30 minutes, taking note of resource usage, node consistency, and the geographical location of of the nodes. This data is stored in a database and used to review the system over time to ensure that it is indeed stable. It is also displayed using a Google maps API to provide an interactive map to users and administrators of where appliances are running.

C. Security Done Easy

Implementing IPSec in the system is complicated because it requires a centralized system to sign a X.509 certificate and



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

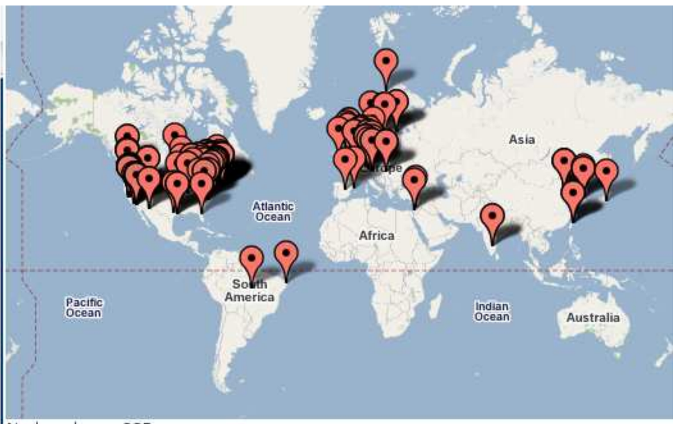
TLB Instruction
Number of sets: 16
Block size: 4096
Associativity: 4
Replacement Policy: LRU

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
```

Fig. 3. A web based AJAX session of SimpleScalar. SimpleScalar is simulator for a MIPS/RISC CPU.



```
Nodes above: 235
Total node count: 321
Ring consistency: 0.9447
Node Avg CPU %: 0.8800
Node Avg MEM: 25762.38
Nodes missing SimpleNode: 0
Last update: Fri Oct 5 10:54:02 2007
```

Fig. 4. The distribution of the current Grid Appliance system. This is one of the many tools administrators have at their disposal.

return it to the requester. To alleviate this problem, the Web interface/management appliance also integrates an interface to facilitate requesting and signing of host certificates for IPsec.

In this approach, the configuration virtual floppy is piggy-backed to hold information needed for users to issue certificate requests for their appliances. The floppy is created automatically at the Web interface appliance and presented to the user for download; it contains the certificate authority's public key and user information registered with the Web front end (e.g. name and email address). Users can then copy these floppies as described earlier to boot appliances of different kinds (worker, client, manager). Upon boot, the appliance checks the virtual floppy and, if necessary, generates a host certificate request for signing. The certificate request is transmitted to the CA automatically, and the appliance polls the Web interface server for updates periodically.

The administrator of the Web interface/management appliance is presented with the list of requestor's credentials. He or she is then given the choice to accept or deny the request. One additional motivation for this approach is that it enables batching of requests; the same virtual floppy containing the CA and user information can be distributed over several computers and booted.

VII. Qualitative analysis

While the appliance has not yet reached a large user base, experience from its usage during the past year has helped provide the feedback guiding some of the design decisions presented in this paper.

Ease of installation: our survey has reassured us that installation is often a simple process, even to novice users (undergraduate and high-school students included). The vast majority of users reported installation times of 30 minutes or less — from installing the VMM to running a demo Condor job. We have been able to demonstrate the system in two hands-on tutorial sessions, where students installed our software from a CD-ROM handed to them on site and submitted jobs to our Grid in less than 30 minutes.

System stability: we rely on the PlanetLab infrastructure to bootstrap our public Grid appliance pool. Because PlanetLab nodes are distributed, highly loaded and faulty, it has proven to be a harsh environment. This has made for time-consuming debugging; on the other hand has served to harden our system considerably and provide us confidence on its stability. We have had a 400+ node wide-area overlay running for several months and logged the execution of tens of thousands of Condor jobs, including a two-week long job batch submitted through a Condor-G entry point from the nanoHUB.

Reaching users: we have had some success in bringing external users to our system. One particular example was a class in Grid computing offered at a Swiss university. The instructor and students used our appliance in two assignments, with appliances deployed on their resources. They were able to do so completely independently from us, relying only on the very limited documentation available at that point. We estimate that a couple hundred distinct appliances have been downloaded and deployed by external users; our anecdotal evidence of where most of our users come from points to the VMware appliance directory site.

Our ability to retain users needs to improve. Our enhancements to allow independent pools aims at reaching and retaining users who are more interested in deploying a local pool than connecting to a shared WAN infrastructure. Furthermore, there are many users who could be potential users but are not aware of the capabilities of a system like Condor — let alone the Grid appliance. Reaching such users is very challenging; our appliance is helpful in providing the ability to quickly demonstrate its basic features through short hands-on self-learning sessions at conferences/workshops.

VIII. Conclusion

The lessons learned in the development and deployment of the Grid Appliances came down to that users involved in grid computing come from varying backgrounds and experiences. Attempting to make a single, slim application in a one fits all method does not work. While making things simpler though, you do not want to make it impossible for more advanced users from feeling comfortable in a familiar environment. Most importantly it is of the utmost importance that a user be removed as a far as possible from system issues and quirks and be able to focus on getting their task done. The Grid Appliance approach is to provide and integrate user interfaces, diversified documentation, and less dependence on centralization. These three cases make handling resources in grid systems significantly less complex from both users and administrators.

Overall, this project has made contributions in reducing the complexity of setting up desktop grids by focusing on the ability for non-expert users to setup and maintain both wide-area and private Condor pools. In this paper we describe our experiences with this system and novel features, which include the ability for Condor clients/managers to self-configure using a DHT and integration with a portable web interface. Other aspects discussed in the paper are not unique to this project but have been integrated in ways that are also described, such as setting up wide area peer-to-peer based IPsec systems and the ability to have a single virtual machine self-configure as a server, worker, or client. The experience and lessons learned discussed in this paper have been obtained by observing and interacting with users; these interactions have helped improve our system and helped shape it into a useful open-source resource available to the community at large.

Acknowledgment

Effort sponsored by the National Science Foundation under grant SCI-0537455. 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 NSF.

References

[1] 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.

[2] C. Sapuntzakis, D. Brumley, R. Chandra, N. Zeldovich, J. Chow, M. Lam, and M. Rosenblum, "Virtual appliances for deploying and maintaining software," 2003.

[3] 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.

[4] 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.

[5] 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

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

[7] R. D. S. Alexander, *RFC 2132 DHCP Options and BOOTP Vendor Extensions*, March 1997.

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

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

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

[11] J. Okajima. (2007, October) Aufs – another unionfs. [Online]. Available: http://aufs.sourceforge.net/

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

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

[17] U. of Wisconsin at Madison. (2007, October) Condor version 6.8.6 manual. [Online]. Available: http://www.cs.wisc.edu/condor/manual/v6.8.6/

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

[19] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, , and H. Yu, "Opendht: A public dht service and its uses," in *ACM SIGCOMM*, August 2005.

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

[21] 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.

[22] 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.

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

[24] A. R. Butt, R. Zhang, and Y. C. Hu, "A self-organizing flock of condors," *J. Parallel Distrib. Comput.*, vol. 66, no. 1, pp. 145–161, 2006.

[25] S. Annapureddy, M. J. Freedman, and D. Mazires, "Shark: Scaling file servers via cooperative caching," in *2nd USENIX/ACM Symposium on Networked Systems Design and Implementation*, May 2005.

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

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

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

[29] 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

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

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

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

[33] 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/