Everlab – A Production Platform for Research in Network Experimentation and Computation

Elliot Jaffe, Danny Bickson, and Scott Kirkpatrick – Hebrew University of Jerusalem, Israel

ABSTRACT

We have pioneered the deployment of EverLab, a production level private PlanetLab system using high-end clusters spread over Europe. EverLab supports both experimentation and computational work, incorporating many of the features found on Grid systems. This paper describes the decision process that led us to choose PlanetLab and the challenges that we faced during our implementation and production phases. We detail the monitoring systems that were deployed on EverLab and their impact on our management policies. The paper concludes with suggestions for future work on private PlanetLabs and federated systems.

Introduction

Evergrow is a European Commission Sixth Framework Integrated Project with around 28 participating research organizations spread across Europe and the Middle East. The project combines research efforts including network measurement [11, 17], distributed systems [1], and complex systems research [9]. Our researchers span the range from systems developers to physicists. At the onset of the project, we realized a need to provide computational and experimental tools for our members. We purchased a set of eight IBM HS20 clusters co-located with some of our research members. Each cluster has 16 blades, where one blade is a storage server and another blade is used for configuration management. We allocated support expenses to the hosting members in order to provide administration, maintenance and support to our users.

Intentionally, the clusters were spread across eight European facilities: Aston University – UK, Universite Paris-Sud 11 "Orsay" (UPSXI) – France, Istituto Nazionale per la Fisica della Materia (INFM) – Rome, Italy, Collegium Budapest Egyesulet (COLBUD) – Hungary, Tel Aviv University (TAU) – Israel, Otto-von-Guericke-Universitat at Magdeburg (UNI MD) – Germany, Universite Catholique de Louvain (UCL) – Belgium and Swedish Institute of Computer Science (SICS) – Sweden. The clusters were deployed in different sites so that we could run "real-world" network experiments using existing wide area network links.

In September of 2004, the team met to decide how to integrate the clusters into a shared resource. It was agreed to setup a VPN between the clusters with a master LDAP server for authentication. We intended to use IBM's GPFS file system to make our storage available throughout the cluster.

For various reasons, our agreed upon approach was not implemented. The reasons included technical problems, networking policies and the availability of local resources. One year later in September of 2005, we undertook a survey of our clusters. We found a very sad state of affairs. Each local administrator had chosen a different stand-alone implementation for their cluster. Initially the clusters had RedHat EL2 installed. Some of the local system administrators changed the operating system to Fedora Core 4, Debian, Ubunto or Mosix. Many of the clusters were inaccessible because the local network policy forbade open access to "internal" computational resources. At one point, we had eight different operating systems. No part of the original plan was universally implemented and hence our researchers had to request permission from each administrator both for a login id and for a firewall exemption so that they could access the nodes.

Figure 1: Geographic location of EverLab clusters.

Our first task was to deploy a monitoring system for all nodes in the eight clusters. We found that Ganglia [10] was easy to install and required only a small number of changes to existing network policies. Ganglia gave us our first view into what the clusters were doing. The results were disappointing. Many of the nodes were idle.

Having realized that our current approach was not working, we started looking at alternatives. One option was to force all the administrators to adopt a standard platform. This was rejected because each domain had their ''standard'' platform, be it RedHat, Fedora Core, Debian or BSD. The administrators did not want to be responsible for an unfamiliar system. The other option was to find a standard platform that could be administered centrally, relieving the local administrators from direct interaction with the installed operating system. Two options were suggested: Grid and PlanetLab. A comparison of these two approaches can be found in [21].

We explored using a Grid infrastructure [3]. Grid systems are designed for computation and could have been deployed across our clusters. Grid environments are reasonably well developed. Such a system would have provided a unified login, the ability to deploy applications across the nodes and a strong monitoring and management infrastructure. The problem was that the Grid is optimized for computation. Applications are automatically deployed to available nodes. A large fraction of our researchers wanted to perform experiments where the location of a process is important. When debugging network experiments, it is important to be able to run test scripts and tracing programs directly on each remote node. The Grid infrastructure does not allow this kind of access. Grid computation nodes are accessible to users only through the Grid management system.

At the time PlanetLab [18] had been deployed to around 500 nodes across the world. PlanetLab supports network experimentation across remote distributed nodes. The system is centrally managed from Princeton University in the United States. PlanetLab itself was not a complete solution for two reasons; first, PlanetLab is designed only for network experimentation. A significant fraction of Evergrow researchers needed computational resources. Secondly, at the time, there were no production level Private PlanetLab installations. At October 2005, we initiated a European PlanetLab workshop in EPFL, Switzerland [16]. We found out there was significant interest at both educational institutions and in industry for implementing and using Private PlanetLabs to share and manage remote resources.

In December 2005, we took up the challenge of implementing PlanetLab on the Evergrow clusters. We called the new system EverLab. The path was treacherous. At the time, the PlanetLab software was not designed for ease of installation. It was a moving target with components being rewritten and upgraded on a regular and unannounced basis. PlanetLab was designed for ''low end'' computers. It ran on single processor from the Pentium family with 512 MB or RAM, a CDROM and 50 GB of local disk and direct connect keyboard connected to a central USB BUS. Our cluster blades have dual 3 Ghz Xeon processors with 4 GB of local ram, 80 GB of disk, no CDROM and a USB keyboard. It took many months to identify the problems and build the appropriately modified kernels and support files.

We succeeded in deploying a Private PlanetLab system. The system provided a centrally managed platform that was usable for experimentation. We installed Ganglia on EverLab so that we could monitor both the old and new systems from one platform. We also developed a custom-built resource reporting system called EverStats. Together, these tools allow our administrators to track system usage and to identify network and hardware problems. Ganglia told us which machines were in use and EverStats told us who used our system and how each node was allocated.

At this point, EverLab was operational and usable by all Evergrow researchers, but it did not yet support High Performance Computation (HPC). To this end, we deployed the Condor [19] system from the University of Wisconsin. We deployed the Message Passing Interface (MPICH2) [5] on top of Condor. With these two additions, Everlab now supports both computation and networking research.

As far as we know, we were the first production-quality Private PlanetLab. Unlike the original Planetlab network which is mainly based on regular PC computers, our network is based on high-end servers with Gigabit Internet connection.

EverLab runs on a subset of the 112 EverGrow nodes. It currently includes more than 50 blades in six clusters. All Evergrow researchers can create an account on EverLab and have quick access to the resources without negotiating with eight separate administrative domains. EverLab is monitored 24x7, and problems are quickly identified. The EverLab administrators can handle system level issues. Local administrators respond to hardware related problems.

The rest of this paper describes the challenges and solutions that we encountered during this journey. We detail the value of Ganglia and EverStats to our administration efforts. There are still many opportunities to improve and extend EverLab some of which are described in the Future Work section. Finally, we conclude with our lessons and opinions about the use of EverLab type systems for new research projects.

PlanetLab

Everlab is based on PlanetLab version 3.2 [18]. This section describes the PlanetLab implementation.

Overall, we found PlanetLab to be a very stable platform once the installation process and initial settings were completed.

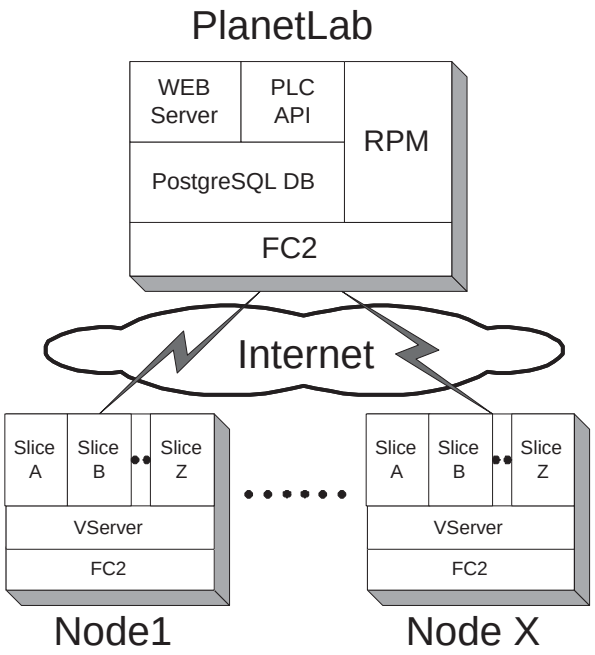

Figure 2: Schematic diagram of the PlanetLab network components.

PlanetLab is a centrally managed collection of distributed computers which are called nodes. The system is designed to be used on a publicly accessible network where all nodes can at a minimum access the central management node. The central management node or PlanetLab Central (PLC) supplies three functions: database for storing system state, web interface for management and a RPM [4] repository for updating the remote nodes. The web interface provides a human readable interface and an XMLRPC interface called PLCAPI for internal use. Remote nodes communicate with the PLC through HTTPS and PLCAPI calls.

At the time of our initial efforts, The PlanetLab Central node ran on Fedora Core 2 (FC2). The PLC used a PostgreSQL database and an Apache web server. Most of PlanetLab was implemented in a mixture of shell and python. The use of common open-source components was a significant factor in our decision to implement PlanetLab. We felt that we could understand, maintain and modify any or all of the components as needed.

Each PlanetLab Version 3 node consists a modified FC2 system. All user activity is performed using virtualization technology implemented by the vserver [8] kernel extension. Node installation is intentionally kept as simple and minimal as possible both to reduce complexity and to increase security. PlanetLab's virtualization unit is called a *slice*. Each slice is a minimal FC2 installation. A user logged into a slice has slice level superuser privileges through the `sudo` command. Slices provide compartmentalization between users and system components, thus reducing or eliminating the possibility of one user modifying or removing a file or component necessary to another user or process.

PlanetLab Security

PlanetLab was designed from the outset as a platform for network experimentation. PlanetLab nodes need free access to and from the Internet in order to provide the broadest possible research opportunities and to limit unexpected network interactions caused by firewalls or local network policies. This focus impacted many of the installation, administration and security aspects of PlanetLab.

PlanetLab utilizes asymmetric encryption keys to create secure authenticated communication channels. These keys are used to identify nodes, servers, and users within the system. There are unique keys for run-time and debug mode operations.

All nodes are assumed to be at risk. Even with the strong compartmentalization provided by the vserver slices, in principle an attacker could enter the root domain and take over the machine. To minimize this exposure and to provide a recovery mechanism from a possible penetration, PlanetLab initially boots from a CDROM. The CDROM contacts the PLC and can either enter a debug mode, boot to the exiting disk based kernel, or re-install the node.

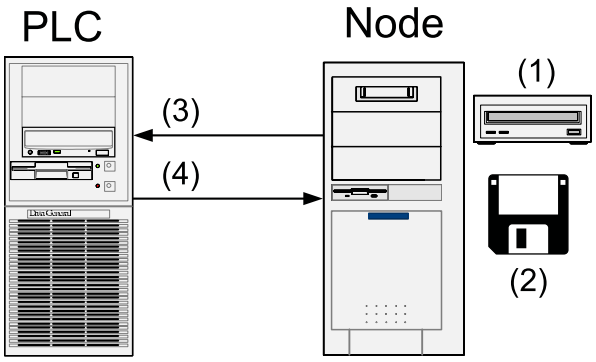

Figure 3: Planetlab boot process. (1) Everlab node boots from CD-ROM. (2) Node gets certificate and identity from floppy drive. (3) Authentication process is done via the PLC. (4) Local files are updated from PLC. (5) Node bootstraps into VServer kernel.

The PlanetLab kernel includes a secure Ping Of Death (POD) implementation which allows the PLC to cause the kernel to reboot given an encrypted secret that only the PLC could have produced.

If a node is suspected of having been compromised, a PlanetLab administrator can cause the node to reboot using the Ping Of Death and to reboot from CDROM into debug mode. At this point, the administrator can log into the node using a special debug

mode SSH key. The administrator can mount the local disk, examine the files and determine if the machine is worth saving. At any point, the administrator can set the nodes status to ''reinstall''. On the next reboot, the CDROM based kernel will wipe the disks clean and install a clean system from scratch.

In the 18 months that we have run PlanetLab nodes on the Internet, we have never had a known penetration. We have used both the POD and the Reinstall option to recover from hardware and software errors.

NetFlow: Security Monitoring and Logging

PlanetLab includes a package called NetFlow on each node. The PlanetLab Netflow component is based on the Netfilter [12] ulogd package. This package tracks all network flows, i.e., communications between this node and all other nodes. The data is available over HTTP from port 80 on each node. The flow traces are very useful in determining which slice was responsible for communication to a given node. This can be helpful when debugging an application or experiment. It can also be used when we suspect that the node has been compromised either by an external party or a rogue experiment.

Installation Issues

PL_BOX

The first efforts to deploy Private PlanetLabs were through a package called ''pl_box:'' PlanetLab in a Box. The package consists of a set of scripts which can download and install all necessary PlanetLab components. The local machine is installed as a PlanetLab Central (PLC) and separate scripts are provided to create deployment CDROM's and kernels. The PLC installation copies the necessary RPM files to a local directory for use when installing PlanetLab nodes. These RPMs include the FC2 package as well as separate PlanetLab packages.

The pl_box package generates all public and private keys, installs the databases and web applications and creates the necessary cron jobs to keep PlanetLab running.

For our implementation needs, there were two major drawbacks to the pl_box system. First, the installed system is a copy of the PlanetLab system. All the web pages, documents and embedded links point back to the original PlanetLab system instead of the newly installed Private PlanetLab. Secondly, there is no upgrade path for pl_box. Changes made to the original PlanetLab system must be manually imported into the private pl_box. At the time, there was no mechanism for change notification and so the public PlanetLab and the private pl_box system were guaranteed to diverge.

Even with these issues, we have found pl_box to be sufficiently stable for our needs. We have had no serious issues since our deployment more than 18 months ago. The PlanetLab project has since replaced pl_box with a new system called MyPLC. The MyPLC system offers the ability to customize the user interface for the private installation. It provides a upgrade mechanism through the use of standard RPM source and binary packages available from the PlanetLab development team. There is no formal upgrade path from pl_box to MyPLC and so we will need to re-install our entire system and re-implement our extensions. None-the-less, we believe that MyPLC represents the future of private PlanetLabs and we plan on upgrading to the new system sometime this year.

Our first challenge was to install pl_box. We chose to use a Fedora Core 4 platform for this purpose. At the time, Thierry Parmentelat at Inria in France showed that it was possible to run a PLC on FC4. We decided to install on FC4, given that it was a fresher, more secure release than the default FC2. The installation and production challenges revolved around changes to core packages such as PHP that pl_box required. Debugging these differences gave us our first understanding of the core functionality.

Once the PLC was installed, we moved to installing new nodes. We started with two local machines that had already been PlanetLab nodes. We had no trouble installing these two machines and were now ready to deploy to the clusters. It was here that our real problems began.

Cluster Ownership

We spent many month prototyping and experimenting with PlanetLab to determine its appropriateness for our installations. At the EverGrow general assembly meeting in December 2005, we presented our results and were given the green light to deploy EverLab on the group's clusters. We then went to each local site administrator and asked for access to deploy the new system. We were met with three types of responses.

Some administrators welcomed us with open arms. We were going to reduce their overhead by managing all of cluster's software and user issues. These systems were converted to EverLab as soon as the existing researchers had finished their ongoing experiments.

Some site administrators had integrated one or more of their cluster's nodes into research workflows. These nodes became dedicated to that project and were unavailable to EverLab. At these sites, EverLab was installed on most but not all of the nodes.

Finally, one cluster never made the transition to EverLab. This group had installed a load balancing version of Linux and were able to keep all of the cluster's node fully loaded more than 90% of the time. It was decided that there was little benefit to be had by moving these nodes to EverLab.

Network Politics

The Evergrow nodes were deployed to eight universities across Europe. Each university had and has

their own network policies. Some of our host universities were already hosting PlanetLab nodes.

We had to fight the network battle at each separate location. PlanetLab minimally requires that:

1. Each node has a public DNS entry.
2. Each node has unhindered access to the PLC.
3. The PLC can send packets to each node for the Ping of Death feature.

In addition, we hoped that all machines would have unhindered access to and from the Internet.

We negotiated with the local system administrators and they negotiated with their local network administrators. In most cases, we were able to get the nodes physically located on a public Internet. The negotiations sometimes required the signatures of university officials or worse, university security officers. All told, it took many months to simply gain access to the remote clusters.

Hardware

The closest cluster to our developers was in Tel-Aviv University (TAU). Unfortunately, TAU has very restrictive access policies and to this day, provides only restricted access to their EverLab nodes. In contrast, The Swedish Institute of Computer Science (SICS) had nodes directly connected to the Internet and were eager to help with our new system. We decided to start testing with the SICS cluster.

Our first challenge was to replace the CDROM based bootstrap process used in PlanetLab nodes. We wanted to maintain the bootstrap ability, but our blades did not have a dedicated CDROM drive or USB drive. We needed to have our nodes boot from the network. We discarded the option of booting from the PLC because of the significant network delays and limited bandwidth between our clusters in Europe and the PLC in Jerusalem. This left the option of booting from a local node.

Each cluster included a management node that was originally designed to provide network boot over DHCP and PXEboot [7]. PXEBoot usually downloads a kernel to the target node. The node boots the kernel in diskless mode and uses NFS to mount the root partition from the boot server. We wanted our root partition to be read-only and to be unique for each node. We could have created a separate read-only directory on the management node and mounted it on each boot node. Instead we choose to incorporate the complete root partition into an initrd file. This file is then downloaded to each node as it boots. The EverLab initrd file contains the complete content of the Boot CD. In a standard system, each PlanetLab Boot CD references a diskette which contains the private keys for that machine. PlanetLab allows an administrator to put these keys into the CDROM itself, thus having a custom CDROM for each node.

Our nodes do not have a diskette, thus we needed a custom initrd for each node. We created a generic initrd and wrote script that copies the generic initrd to a custom file and installs the private keys. This script is then run once for each target node on the local cluster.

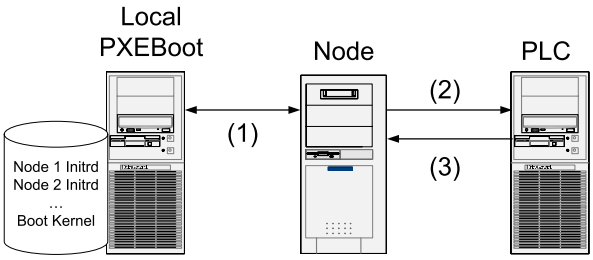

Figure 4 Everlab PXE boot solution. (1) Boot kernel and initrd downloaded from local PXE boot server via PXEBoot. (2,3) Authentication process is done via the PLC and local files are updated. Finally, node bootstraps into VServer kernel.

The blades have two Ethernet Network Interface Cards (NICs). We choose to use one NIC for the bootstrap process and the second for the public Internet. We use the private NIC only during the boot process and leave it un-configured during normal EverLab operations. Slices are unable to configure the NICs. The main benefit is that EverLab users have no way to attack or even to see the bootserver. We believe that this increases the probability that the boot image remains intact. Our approach is not as good as a read-only CDROM, but we feel that it strikes the right balance between security and complexity for our system.

Once we had the PXEboot and initrd process working, we were able to boot the default PlanetLab kernel. Unfortunately, our blades were newer than the supported PlanetLab nodes. The running nodes had no network drivers and no keyboard controller. Each blade has a USB keyboard, but the default PlanetLab nodes did not install the appropriate drivers. Similarly, the blades used a network card that was not available when FC2 was first released. We were left with a node that was clearly running something, but that was deaf and blind.

Through trial and error we were able to identify the appropriate drivers and configuration files and to rebuild our initrd files. This process took the better part of a month, but we were finally able to debug and boot our nodes.

Management Challenges

Our intention was to develop a system that could be managed remotely. The PlanetLab system provided most of that functionality. PlanetLab defines four types of capabilities:

1. *Administrators* These users can perform all PlanetLab operations including Ping of Death and re-install. They can enable or disable features and capabilities for other users.
2. *Principal Investigators* These users are responsible for the use of a set of nodes. They can

enable or disable access for their students and can create slices.
3. *User* Can deploy a slice to one or more nodes and can log into those nodes.
4. *Tech* These users can administer their local site, performing admin like functions only on those nodes.

We use the standard PlanetLab definitions, but allocated each of the local system administrators with Principal Investigator and Tech capabilities. These system administrators are then asked to enable accounts for researchers in their institutions. Researchers that do not have a local cluster administrator are managed by the central EverLab administration team.

Hyperthreading Performance Issues

During the decision process, our computation based researchers wereconcerned that the PlanetLab infrastructure would require a significant fraction of each node's CPU cycles. We were able to show that on our hardware, the difference between a program running on a stock kernel and one running in a slice under PlanetLab was less than 3%. This experiment was not particularly scientific, but it was sufficient to assuage the fears of the HPC researchers.

Our nodes came with two Intel Xeon 3.06 Ghz processors that support Intel's Hyper-threading Technology (HTT) [22] . In theory Hyper-Threading Technology should provide a performance boost of up to 30%. We found that this was true only on heavily loaded server systems running a number of CPU-intensive processes that is larger than the number of installed physical CPUs. In our workloads, we typically have only as many compute tasks as CPUs.

The Linux kernel views each hyper-threaded processor as two virtual processors. When a task is runnable, it is assigned to one of the virtual processors. Ideally, since we have two physical processors per node, the kernel should schedule each task to a different physical processor. Unfortunately, we found that in many instances, the kernel scheduled both CPU intensive tasks to virtual processors on the same physical processor. The resulting cache misses and contention significantly reduced our systems performance.

In light of this finding, we have turned off Hyper-threading on most of our nodes.

Stability

IBM describes the HS20 cluster as: 'Powerful 2-way Intel processor-based blade server delivers uncompromising performance for your mission-critical applications.'' [6] We purchased this equipment in 2003/2004 from IBM because we valued IBM's reputation for reliability and service. We have since had cause to regret this decision.

Each of our blades came mounted with two Toshiba MK4019GAXB [20] 40 GB disk drives. These drives are 2.5'' hard disk drives of the type found in many laptops. IBM probably choose to use 2.5'' drives because of the size restrictions imposed when fitting two drives on each blade.

In the interest of reducing disk management, each PlanetLab node mounts its own local disks as a single large logical volume. This construct enables the system to allocate disk space without concern for the size of each partition or disk drive. The drawback is that if any of the drives fail, the whole partition is corrupted and lost. Recall that in PlanetLab, it is easy to re-install a node. The loss of any one node is pretty trivial and the resources can be easily recovered (although the data on that node is lost).

At installation time, the Evergrow clusters had a total of 244 MK4019GAXB drives. In the first three years of ownership, we have replaced more than 50 of these drives. By far, the most common problem and the largest management headache has been the failure of disk drives. We have not identified any common cause behind these failures. Some of our clusters are in very professional data centers with significant cooling capacity and managed power. Other clusters are in less professional locations with limited cooling and whatever power is available from the wall socket. Location does not seem to be a factor in the failures. We can only conjecture that these drives were defective in one way or another. There have been no failures with any of the non-MK4019 replacement drives.

Monitoring

Monitoring of resources is an important part of any research project. At a minimal level, monitoring tools identify software, systems and hardware that may not be operating as expected. One level up, monitoring provides a list of assets that administrators and users can reference to identify and find available resources. At the management level, the project coordinators can watch the monitoring systems to identify users that are utilizing the projects resources. This data also delineates those registered users who are not using the system.

Ganglia

We began to wonder about utilization of our new system about six months after our partners received their cluster hardware. We knew that each partner had deployed the systems, but we had no idea if the clusters were in use or even if they were operational. Our very first effort was to implement a centralized monitoring system using the Ganglia [10] software package. Ganglia provided at-a-glance status for each cluster and each node.

Ganglia implements both push and pull communications. Each monitored node runs a ganglia daemon called gmond which intermittently collects information about the local system state. Each gmond daemon sends a regular update to a gmeta collector daemon. The gmetad daemons maintain a list of all clients that

sent it data along with the details for those clients. We have one gmetad daemon for each cluster and a separate one for the EverLab system. In our implementation, a central gmetad daemon running on our main server polls each of the gmetad daemons on a regular basis and collects a snapshot of that daemons stored status. The collected data is then stored in RRD [13] databases for graphing and presentation.

We keep a web browser open to our local Ganglia monitor. At a glance, we can see which clusters are operational, busy, or down. Ganglia provides summary statistics of Total, Up and Down hosts which quickly reflect the overall system health.

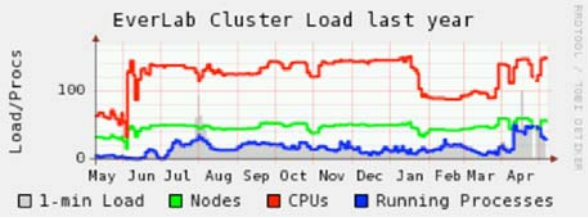

Figure 5: May 2006 to May 2007 EverLab One Minute Load Average.

The main Ganglia load graphs display the 1 minute load, the number of nodes, the number of CPUs and the number of running processes. An optimally utilized system would have one process for each CPU. The Everlab One Minute Load Average graph shows the 1 minute load on the EverLab system between May 2006 and May 2007. The line at the top of the graph shows the total number of CPUs, some of which are actually virtual hyperthreaded CPUs. The second line shows the number of reported nodes and the third line shows the number of running processes.

As can be seen, the number of nodes changes over time. The majority of these outages are related to power and cooling problems in the remote data centers. The graph also shows the growth in EverLab usage. Initial use was minimal until after the EverLab Workshop in June of 2006. From that point until March 2007, we averaged about one process for every two nodes. Toward the end of this period, we saw usage increase to approximately one process per node.

Ganglia has been very useful in Evergrow and EverLab, but was not particularly successful in the PlanetLab environment. In 2001, one of the Ganglia developers joined the PlanetLab project and began deploying Ganglia over PlanetLab. Over time, the Ganglia installation was removed and forgotten. It seems that Ganglia did not provide enough benefit to the PlanetLab team to warrant its maintenance costs.

We believe Ganglia was appropriate for EverLab but inappropriate for PlanetLab for of the following reasons:

1. EverLab is naturally organized by clusters of nodes. The gmond daemons communicate over the local network to their gmetad parents. In PlanetLab, there is no natural structure and hence each gmond must send remote messages to a centralized gmetad. This increases the rate of lost message and requires significant bandwidth at the central node.
2. PlanetLab nodes have traditionally been heavily utilized. It is very rare to find a PlanetLab node that has no running processes. Ganglia shows that their nodes are busy. Ganglia provides dozens of detailed metrics, but does not differentiate between slices. From a high level perspective, We use Ganglia only to show that a node is busy or down. The detailed metrics have not been useful in our environment.
3. PlanetLab has more than 600 nodes. Ganglia does not scale particularly well over a few hundred nodes. For example, in Evergrow, the Ganglia web front end periodically queries the main gmetad daemon for the system state. The data is returned as a single large XML document at least 220K bytes in length. The equivalent file for PlanetLab would be more than 1 MB large. Sending this data over the wire every 60 seconds is inefficient, particularly since most of the data has not changed.

We are very pleased with Ganglia for our size installation and would be likely to choose it again. For us, the benefits of a simple monitoring system outweigh the network traffic overhead. Ganglia is a rough tool. The instantaneous data is frequently incomplete because of the way that the gmetad daemon collects and updates its internal data structures. Improvements in this area would make Ganglia more useful and reliable.

EverStats

Ganglia provided our system with cluster and host level monitoring. We could identify node status and utilization. But our project administrators wanted more. They asked us about the users and their projects. Which projects were deployed on EverLab? Were the projects computationally focused or more experimental? How many unique projects were actually using the system? To address this issue, we developed the EverStats usage monitoring system.

Our first challenge was to collect data from the EverLab nodes. We knew that the CoMon [15] project had developed a tool on PlanetLab to monitor node usage, so we borrowed their underlying monitoring tool called slicestat [14]. The slicestat daemon runs on each PlanetLab node and like the gmond daemon collects performance metrics. Slicestat has the added benefit that it understands the PlanetLab virtual server architecture and can report data according to each slices activity.

The CoMon project polls each slicestat and provides current CPU data along with 1 minute and 15

minute network statistics. In many ways CoMon overlaps with Ganglia as a monitoring tool.

Our interest was not in the short term node and slice status, but in the historical usage patterns. We installed slicestat on all EverLab nodes and then built a custom tool called EverStats to poll the deamons and store the results in a long term database. EverStats provides summary reports on usage for the past week, month and year and allows administrators to drill down from both a slice or node view.

In order to minimize network traffic, EverStats polls the slicestat daemons once every five minutes. We keep the raw data for 24 hours and then summarize it as daily data in our database. Short term activities such as running a program for 30 seconds are unlikely to be visible on Everstats. On the other hand, computation or experiments that run for a hour or more will certainly be reported.

There are more than 65 unique projects registered in the EverLab database. New users tend to create a test slice to familiarize themselves with the system. These users then move to a project slice that is shared by their research team.

The EverLab slice groups report reports on the cumulative data for all slices in each defined group. The sample EverStats Slice Group Report shows a representative report from April 2007. The System group represents the basic EverLab services. These services tend to be active for short periods of time, but are visible when a node is idle.

The Condor slice group represents the distributed Condor instantiations on each of our nodes. Condor is currently under very light load and hence the values represent a sort of steady state overhead similar to the System group.

The rest of the groups represent research originating at their respective universities. The ‘‘Others’’ group is a catch-all for research from one of our non-cluster partners. There are currently seven slices in the ‘‘Others’’ group. The university research groups have between two and five slices each.

As can be seen from the report, most research projects do not use the full power of EverLab. As the number of nodes in an experiment increases, so too does the complexity of deployment, debugging and monitoring. Projects tend to use the maximum number of nodes necessary to produce a reasonable academic paper.

Of the six project groups, more slices are CPU bound with CPU loads running between 88.44% and 261%. A factor over one indicates that there is more than one process running concurrently in these slices. Experienced High Performance Computing researchers attempt to allocate exactly one process to each processor in order to decrease contention for CPU cycles

The ‘‘Others’’ group provides a good example of slices involved in networking experiments. These slices are generating on average 45Kbps of incoming and outgoing traffic over the life of the experiments. As can be seen, the System and Condor groups are minimal users of both CPU and Networking.

EverStats is a useful tool in its current instantiation. Potential extensions include the graphing of trends and improvements in the sampling technology. Graphs and Trend analysis would be helpful for presentations and for tracking the natural growth and decline of project activities.

The current sampling technology serially queries each of the nodes. In addition to being inefficient, this approach takes a significant fraction of the five minute query period. While acceptable for EverLab, the serial query mechanism takes more than thirty minutes on the full PlanetLab for each query cycle.

Ideally, we would like to see EverStats integrated into the base CoDeen and CoMon projects for use by both Private and Public PlanetLabs.

Education

The Evergrow project is composed of researchers in Computer Science and Computational Physics. All of our researchers are computer literate and have some scientific programming ability. At the beginning of the Evergrow project, we assumed that researchers would use any and all computational resources that we could provide. In fact, we found that computational resources are currently widely available. Desktop workstations have enough processing power to handle many tasks previously allocated to dedicated processors. One of our

Slice Name	Nodes	Total CPU Hours (all nodes)	Average % CPU	Avg. Outgoing Bandwidth (Kbps)	Avg. Incoming Bandwidth (Kbps)
System	52	2129.15	0.72	0.04	0.06
Condor	52	1086.05	0.96	0.21	3.40
Aston	34	19048.40	97.23	0.00	0.00
HUJI	25	5738.09	96.75	0.01	0.67
Orsay	10	2729.79	261.93	0.00	0.00
SICS	19	1575.53	88.44	2.82	1.14
UCL	25	4071.05	134.48	18.18	1.61
Others	52	5088.41	8.32	44.70	46.69

Table 1: Sample EverStats slice group report.

authors processes gigabytes files on his laptop. The performance is not great, but the benefits of taking your work with you outweigh the time to completion.

Our partners provided a list of requirements when we started the Evergrow project. Some wanted High Performance Computation (HPC) services. Others wanted distributed network platforms for experimentation. The resulting EverLab system provides both, but we found that our partners needed help getting up to speed. Our most effective tool has been hands-on workshops. Our first workshop was held in June, 2006.

PlanetLab (and EverLab) present the world as set of virtual servers running on remote hosts. At one time, using telnet, SSH and X-windows was the standard method for interacting with remote hosts. Today, undergraduate and graduate students use the Microsoft Windows platform and Microsoft Remote Desktop connection. PlanetLab's interfaces are much more basic and are less familiar to many of our partners.

During our workshop, we walked the participants through the EverLab process. To get started running your own code on EverLab (or PlanetLab), a researcher must:

1. Have a registered site and Principal Investigator (PI). We created a pseudo-site called EverLab for all of our users.
2. Request an account by filling out a web form.
3. Wait for the account to be enabled by the PI or site administrator.
4. Create and upload an SSH key to the management web site.
5. Have the PI create a slice for your project.
6. Assign the users to the new slice.
7. Assign the slice to one or more nodes.
8. Wait until the slice propagates to the target nodes.
9. Log into the slice using SSH.

The total latency from start to finish is minimally about one and a half hours. For a user trying this remotely, it can take between one to three days just to be able to log into the nodes. The major benefit from our workshop was to shorten this initial period and to get users working during workshops' first day. The second day was spent learning about Ganglia, Condor and custom deployment scripts that other researchers have written for deploying experiments on PlanetLab/EverLab. Summary of presentations and tutorials are available on the web [2].

We have found that all of our active researchers attended our workshop. It may be that other researchers do not need our dedicated resources, or that the learning curve is too steep. We plan on continuing our educational efforts and working with our researchers to identify the barriers to better system utilization.

Future Work

We have identified a number of areas for future work on PlanetLab in general and in particular on EverLab. Each of these areas are extensions of our experience with the current EverLab system and its user community.

Security

Fedora Core 2 (FC2) was first released in May 2004. Fedora Core 4 was the more recent release as of September 2005, when we started working on PlanetLab. As of the summer of 2006, EverLab is still running FC2 on its nodes and FC4 on its central management node (PLC).

Fedora Core 2 officially reached its end-of-life in June 2007. Fedora Core 4 reached its end-of-life in January 2007. Fedora Core 5 was retired in July of 2007. The implication is that bug releases and security patches for these systems are no longer available from the Fedora team for these systems.

Our experience and that of the PlanetLab Consortium is that there have been almost no security issues related to FC2 or FC4. The latest PlanetLab V4.0 release still supports only FC4. Common wisdom would suggest that we update to FC7 as soon as possible. Our experience has been that our deployed version of the three year old FC2 has been stable and secure and that there is little urgency to upgrade.

Usability

Our user community differs from the standard PlanetLab community in their grasp of UNIX tools. The PlanetLab community includes many systems researchers who understand the Linux operating system and its user level tools in detail. Our community of physicists and computer science theoreticians do not have this level of systems knowledge. We would like to see future systems include a suite of basic, easy to use tools for accessing the nodes, deploying applications, collecting logs and monitoring the experiments activity. In most cases, the effort to develop these tools is one of packaging and documentation.

Improved Coordination

With the release of PlanetLab V4, there has been significant improvement in the installation and upgrade processes for private PlanetLabs. The major remaining issue is coordination on PlanetLab changes. We would like to see a PlanetLab Engineering Task Force (PETF) that would manage platform changes and coordinate platform security.

We see PlanetLab as a moving target. There are many possible ways to extend and improve the system. The challenge is to choose the appropriate changes for the private PlanetLab community. Private PlanetLabs value stability and security over experimental features. The PETF would collect and document these changes. It would provide a repository for all blessed changes and versions of the system.

As a production system, PlanetLab should have a security coordinator. The PETF would track published and zero-day attacks on PlanetLab or its constituent

components. It would provide timely notification and updates to administrators concerning these attacks and would coordinate efforts to detect and correct these problems as they occur.

The PETF could organize workshops and conferences for Private PlanetLab administrators and users as a way to educate the community and to identify areas for improvement and growth.

Federation of PlanetLab's

One vision presented by the PlanetLab community is to integrate remote PlanetLabs in a federation. Users on one system would be able to utilize resources on federated systems while abiding by inter-system usage policies. This concept requires coordination of the detailed federation interfaces as well as the definition of appropriate system level policies.

The EverLab installation has many of the features required for a federated PlanetLab system. We operate in a production environment and our system is not oversubscribed. While our project would welcome federation with other private PlanetLabs, our partners network administrators would be hard-pressed to open the system to non-partner sites without additional controls to protect their networks from abuse.

Conclusion

Everlab serves as a model for future research efforts. It bridges the gap between Grid based HPC installations and free-for-all experimentation systems. EverLab makes efficient use of administrative resources and provides reporting services to monitor system usage, reliability and responsiveness. EverLab provides a service for setting policy on resources so that all participants have access to the shared resource. With the exception of pure HPC projects, We believe that future efforts should include an EverLab style system for system management, resource allocation and monitoring.

Acknowledgement

We would like to thank the PlanetLab Consortium based at Princeton for its extensive support and encouragement, without it the EverLab project could not have been accomplished. Special thanks to Consortium members Marc E. Fiuczynski and Steve Muir. Lior Ebel was instrumental in developing EverStats.

Author Biographies

Elliot Jaffe received his B.Sc. in Mathematics from Carnegie Mellon in 1985. He worked in industry as a system administrator, developer, integration specialist, manager and CTO. He returned to academia and received his M.Sc. in Computer Science from The Hebrew University in 2005 where Elliot is currently working towards his Ph.D. in Computer Science. Elliot's research interests are in the areas of Software Engineering, Distributed Systems and Large Scale Storage.

Daniel Bickson is currently a Doctoral candidate at the School of Engineering and Computer Science at the Hebrew University, Jerusalem, Israel. His research interests include Communications, Network Security, Distributed Systems and Belief Propagation.

Scott Kirkpatrick has been a Professor in the School of Engineering and Computer Science at the Hebrew University, Jerusalem, Israel since 2000. While at IBM Research in his previous career, he compiled a distinguished scientific record (90+ publications, 10+ patents) and was elected a Fellow of the AAAS, the APS and the IEEE. Some of his papers are among the top cited of all time, in both Physics and Computer Science. In addition, Prof. Kirkpatrick has more than 20 years of experience in management at IBM, supervising several large multi-team research and development projects, and currently coordinates the 25+ Partner FP6 IP EVERGROW (2004-2008).

Bibliography

[1] Bickson, D. and D. Malkhi, "The Julia Content Distribution Network," *The 2nd USENIX Real World Distributed Systems (WORLDS '05)*, 2005.

[2] Everlab Workshop, Huji, Jerusalem, Israel, June 7-8, 2006, http://www.cs.huji.ac.il/labs/danss/p2p/evergrow-workshop .

[3] Foster, I., C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal High Performance Compututing Applications*, Vol. 15, Num. 3, pp. 200-222, 2001.

[4] Foster-Johnson, E. (Red Hat), *Red Hat RPM Guide*, March, 2003.

[5] Gropp, W., "Mpich2: A New Start For MPI Implementations," D. Kranzlmüller, P. Kacsuk, J. Dongarra, and J. Volkert, editors, *PVM/MPI*, Vol. 2474, *Lecture Notes in Computer Science*, p. 7, Springer, 2002.

[6] IBM, *IBM Blade Servers – Bladecenter T-HS20 Server*, http://www-03.ibm.com/servers/uk/eserver/bladecenter/hs20/more_info.html .

[7] Intel, *Preboot Execution Environment (PXE) Specification*, 2002.

[8] Ligneris, B. D., "Virtualization of Linux-Based Computers: The Linux-vserver Project," *19th International Symposium on High Performance Computing Systems and Applications, HPCS 2005*, pp. 340-346, May, 2005.

[9] Lukic, J., E. Marinari, O. C. Martin, and S. Sabatini, "Temperature Chaos in Two-Dimensional Ising Spin Glasses With Binary Couplings: A Further Case For Universality," *Journal of Statistical Mechanics*, 2006.

[10] Massie, M. L., B. N. Chun, and D. E. Culler, "The Ganglia Distributed Monitoring System: Design, Implementation and Experience," *Parallel Computing*, Vol. 30, Num. 7, July, 2004.

[11] Morato, D., E. Magana, M. Izal, J. Aracil, F. Naranjo, F. Astiz, U. Alonso, I. Csabai, P. Haga, G. Simon, J. Steger, and G. Vattay, ''The European Traffic Observatory Measurement Infrastructure (ETOMIC): A Testbed For Universal Active And Passive Measurements,'' *TRIDENTCOM '05: Proceedings of the First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities (TRIDENTCOM'05)*, pp. 283-289, IEEE Computer Society, Washington, DC, USA, 2005.

[12] Napier, D., ''IPTables/NetFilter – Linux's Next-Generation Stateful Packet Filter,'' *SysAdmin: The Journal for UNIX Systems Administrators*, Vol. 10, Num. 12, pp. 8-16, Dec., 2001.

[13] Oetiker, T., http://ee-staff.ethz.ch/~oetiker/webtools/rrdtool/ .

[14] Park, K. and V. Pai, http://codeen.cs.princeton.edu/slicestat/ .

[15] Park, K. and V. S. Pai, ''Comon: A Mostly-Scalable Monitoring System For Planetlab,'' *SIGOPS Operating Systems Review*, Vol. 40, Num. 1, pp. 65-74, 2006.

[16] Second European Planetlab Workshop, EPFL, Laussane, Switzerland, October 27-28, 2005, http://lsirwww.epfl.ch/planetlabeverywhere/ .

[17] Shavitt, Y. and E. Shir, ''Dimes: Let the Internet Measure Itself,'' *Computer Communication Review*, Vol. 35, Num. 5, pp. 71-74, 2005.

[18] Spring, N., L. Peterson, A. Bavier, and V. Pai, ''Using Planetlab For Network Research: Myths, Realities, and Best Practices,'' *SIGOPS Operating Systtems Review*, Vol. 40, Num. 1, pp. 17-24, 2006.

[19] Thain, D., T. Tannenbaum, and M. Livny, ''Distributed Computing in Practice: The Condor Experience: Research Articles,'' *Concurrent Computing: Practice and Experience*, Vol. 17, Num. 2-4, pp. 323-356, 2005.

[20] Toshiba, *Mk4019gax*, http://www.sdd.toshiba.com/main.aspx?Path=81820000000700000010000659800001516/818200000aff000000010000659c000026ad/8182000001920000000010000659c0000279f/81820000019f000000010000659c0000054e .

[21] Ripeanu, Matei, Mic Bowman, Jeffrey S. Chase, Ian Foster, and Milan Milenkovic, ''Globus and PlanetLab Resource Management Solutions Compared,'' *Proceedings of the Thirteenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-13)*, Honolulu, Hawaii, June, 2004.

[22] Intel, *Hyper-Threading Technology*, http://www.intel.com/technology/platform-technology/hyper-threading/index.htm .