SLAC–PUB–9193
April 2002

A General Purpose High Performance Linux Installation Infrastructure*

Alf Wachsmann
Stanford Linear Accelerator Center (SLAC),
Stanford University,
Menlo Park, CA USA

Abstract

With more and more and larger and larger Linux clusters, the question arises how to install them. This paper addresses this question by proposing a solution using only standard software components. This installation infrastructure scales well for a large number of nodes. It is also usable for installing desktop machines or diskless Linux clients, thus, is not designed for cluster installations in particular but is, nevertheless, highly performant.

The infrastructure proposed uses PXE as the network boot component on the nodes. It uses DHCP and TFTP servers to get IP addresses and a bootloader to all nodes. It then uses kickstart to install Red Hat Linux over NFS.

We have implemented this installation infrastructure at SLAC with our given server hardware and installed a 256 node cluster in 30 minutes. This paper presents the measurements from this installation and discusses the bottlenecks in our installation.

1. Introduction

The problem of (re–)installing a lot of Linux machines in a very short amount of time is getting more difficult with the number of machines to be installed.

Two examples for large Linux cluster installations: Many dot.com companies have several thousand Linux machines in their WEB server clusters. All four experiments for LHC at CERN are proposing Linux farms with several thousand machines.

The EU DataGrid Project [1] even has a dedicated work package "Fabric Management" [2] to further investigate how to install and maintain large clusters.

The difficulty is that many tools and toolsets exist which try to solve this installation or upgrade problem. But most if not all of them assume that the hardware is not diverse and are using some sort of disk cloning techniques (e.g. Systemimager [12]). This approach does not work for farms with heterogeneous hardware and makes it necessary to have one installation infrastructure for homogenous farm nodes and one for more diverse machines.

Other tools like IBM's LUI [5] are dealing with this hardware diversification. But getting (e.g.) all desktop machines configured for a larger site is a nightmare.

Red Hat's installer "anaconda" is ideal for both of these situations because it probes the hardware and installs the necessary driver and software packages automatically. "Anaconda" is automatically executed from "kickstart", Red Hat's automatic installer program.

The question arises whether this rather general installation approach does scale for a large number of machines to be installed or whether it really makes sense to have a special cluster installation infrastructure.

One other objective of our approach is to use only servers and services which are already in place or can be reused for other purposes. Unlike an "rsync server" which can only be used for "systemimager" (to pick only one example), a DHCP server for the farm node installation can also be used for installing desktop machines. Furthermore, one particular system image – to stay with the "systemimager" example – can only be used on exactly the same hardware. Whereas a set of RPM ("Red Hat Package Manager") packages used by "anaconda" is usable for all Linux installations.

NPACI's "Rocks" [9] package contains such a general purpose installation infrastructure but some manual intervention is still necessary and their approach re–installs a machine whenever it boots. We would like to pursue a less radical solution. Furthermore, "Rocks" is meant to install only small clusters up to 128 nodes [8].

* Work supported by Department of Energy contract DE–AC03–76SF00515.

Given all this, the design goals for our Linux installation infrastructure are the following:

1. Use only standard services which are in place already or can be used for other purposes.
2. Make it highly scalable to allow installation of thousands of machines at the same time.

This paper describes our solution and gives some benchmark results for our implementation to show its feasibility.

In the next section we will describe all components for our infrastructure. Section 3. discusses the scalability features of this infrastructure and Section 4. presents the measurements we made installing up to 256 nodes at the same time.

2. Architecture of Installation Infrastructure

Our architecture for installing large Linux clusters facilitates the network boot capability (Preboot Execution Environment PXE [6]) of modern network interface cards (NIC's). This requires DHCP and TFTP servers to get the necessary programs and configuration files to the clients. The chain of programs loaded and started from PXE are: a bootloader, a Linux kernel, and an initial RAM disk for this kernel. The Linux kernel starts kickstart to install Red Hat Linux. After Linux is installed, a post–installation process makes necessary local adaptions.

Figure 1 shows all necessary servers for the proposed architecture in its most general appearance.

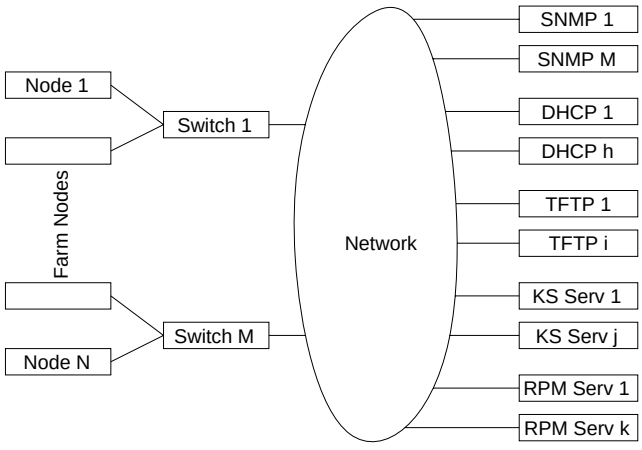

Figure 1: General Architecture of the Installation Environment

In the following sections we will describe all used components in more detail. And we will present some glueing scripts which bind everything together to make a mass installation entirely automatic and unattended.

The complete technical details and all mentioned scripts are available from [13].

2.1. Preboot Execution Environment (PXE)

To boot an Intel PC over the network, most NICs come equipped with PXE, a standard proposed by Intel. The first step in this procedure is a DHCP broadcast from a client to get its IP address. It allows a PXE server to send the IP address and a menu back to the client such that a user can choose which operating system should be installed on this system. PXE on the client then loads a bootloader via TFTP which in turn loads the chosen operating system. We only need the DHCP and TFTP part to get an IP address and a bootloader onto each machine. We always want to install Linux as the OS. Hence, we don't need a PXE server, only a DHCP and TFTP server.

2.2. DHCP

For several reasons each machine at SLAC needs a **static** IP address. Thus, we cannot simply provide a range of free IP addresses to our DHCP server which would be assigned to clients in chronological order of their requests.

To allow static IP address assignment via DHCP, the DHCP server needs to know all client's MAC addresses beforehand. This is no problem for already running machines which need to be reinstalled. For new machines with unknown MAC addresses there are several possibilities to achive this. 1. Get a list of MAC addresses and machine serial numbers from the vendor and keep track of each machine during racking. 2. All machines have barcode labels which contain their MAC address. After the machines are racked, they can be scanned in their racked order. 3. Switch the machines on in the order you want IP addresses assigned to them. Scan the `dhcpd` logfiles for incoming requests and assign IP addresses according to this order. This third option is chosen by NPACI for "Rocks" [9].

All this is painful, needs additional effort and/or hardware and is error prone. The third approach highly depends on the timing of the machines when they are powered on.

2.2.1. MAC address Detection

We have developed a script to detect MAC addresses automatically and on–the–fly while the clients perform their PXE DHCP broadcast. This broadcast sends a signal over the Ethernet through the switch to find a DHCP server. The switch sees this new MAC address on one of its ports and fills its "bridging table" with this information. Our switches, Cisco Catalyst 6509 switches, can be queried via SNMP to read out the bridging table. IP addresses can now be assigned in the order of ports on the switch: the machine with the MAC address seen on the first port gets the first IP address and so on.

The only prerequisite for this technique to work is careful Ethernet cabling between machines and switch. The machine in the first position in the rack must be connected to the first port on the switch, the second position in the rack must be connected to the second port, and so on. This order determines the order in which IP addresses are assigned to the machines.

Our Perl script which queries the switch, takes as input only the switch name, the cards in the switch and the ports on these cards to look for. It outputs a new DHCP server configuration file (`dhcpd.conf`) with all previously unknown MAC addresses added. A second script observes this `dhcpd.conf` file and stops and starts the DHCP server whenever the configuration file has changed.

This entire process is highly time critical because the PXE standard requires clients to issue only 2 DHCP requests. Therefore, our script reads the entire `dhcpd.conf` file only once at the beginning and keeps it in memory. Whenever it adds one or more MAC addresses, it writes out the entire file instead of editing it. All changes to this file made in the meantime by other scripts would be overridden! We will come back to this point in Section 2.4.2.

With this programming trick and our clients, which issue about 10 DHCP requests which take about 1 minute to eventually time out, we have enough time to detect the MAC addresses automatically.

2.3. TFTP

PXE poses some requirements on the TFTP server it can use. For details about the special requirements, see H. Peter Anvin's discussion in [4]. One open source server which meets these conditions is the TFTP server by Jean-Pierre and Remi Lefebvre [7]. This TFTP server is able to run as a standalone daemon which improves performance compared to a TFTP server started by an `inetd` process.

2.4. Bootloader

The bootloader program is loaded from PXE via TFTP to the client and then started. We are using H. Peter Anvin's bootloader `pxelinux.0` from his `syslinx` package [4]. It loads its own configuration file via TFTP and executes whatever this configuration file dictates.

2.4.1. Bootloader Configuration

If the bootloader configuration file tells the bootloader to perform a network boot, it loads (in our case) a Linux kernel and its inital RAM disk over the network onto the client. The bootloader then starts the Linux kernel with all necessary parameters which it finds in its configuration file. These parameters tell the kernel to perform a kickstart installation and to look for the kickstart configuration file in a certain location on an NFS server.

In case its configuration file tells it to perform a boot from the local hard disk, it does just this.

The `pxelinux.0` bootloader allows us to have a separate configuration file for each client. The filename is the machine's IP address in hexadecimal notation without any dots. This feature is handy for preventing endless re–installations.

2.4.2. Preventing Endless Re–Installations

After the OS installation is complete, the machine reboots. Because the BIOS is set to PXE boot, we need a mechanism to prevent a new installation. The obvious solution, changing the DHCP server configuration to not give the client a filename for its following TFTP phase does not work in our situation. See Section 2.2.1. for the reason.

`pxelinux.0` starting with version 1.53 understands a "`localboot`" parameter in its configuration file. This parameter tells the boot loader to start the boot process from the local hard disk and, thus, prevent a new network installation. All we have to take care of is to change the configuration file for each client after it is done installing. We transfere one additional (empty) marker file via TFTP at the end of the kickstart phase. This adds an entry in `syslog` on the TFTP server. With this additional file transfer it is simple to write a script which looks in the `syslog` log file and checks for the transferred marker file and changes the pxelinux

configuration file for the machine which received this file.

Starting a mass installation, each machine's bootloader configuration file is symlinked to a file allowing it to perform a network boot. The above described mechanism removes this symlink when the machine is done and creates a new one pointing to a file telling the machine to perform a boot from its local hard disk.

2.5. Kickstart over eth1

Our client machines have two NIC's: "eth0" is on the motherboard and "eth1" is a PCI card. Because the motherboard NIC cannot perform a PXE network boot, we have to connect the machines via "eth1." The production version of Red Hat Linux at SLAC is still 6.2. The Red Hat installer "anaconda" coming with this version does not allow us to select a specific Ethernet adapter to perform the installation – it is hardcoded to "eth0". Hence, we had to change this to "eth1" in the "anaconda" source code, compile it, and generate a new initial RAM disk (`initrd.img`).

"Anaconda" in Red Hat 7 and later has a new parameter "`ksdevice`" for exactly this purpose.

2.5.1. NFS Mounting

Kickstart can do the installation either from a locally attached media (harddisk or CD-ROM) or over the network using FTP, HTTP or NFS. We have a centrally maintained mirror of a (locally modified) Red Hat distribution in NFS. Kickstart mounts this filesystem and gets all necessary software packages from there. The kickstart configuration file determines the NFS server name and the filesystem to mount.

2.5.2. %post

The last step in a kickstart installation is to execute the "%post" section from the kickstart configuration file. We are using this step to switch off unneeded services and install our real post–installation script.

3. Scalability

Primary design goals for this installation infrastructure are scalability and performance. For smaller sites, scaling down is an issue as scaling up is for larger sites. Our infrastructure meets both these needs.

3.1. Scaling Down

All servers (DHCP, TFTP, NFS) and all control scripts can run together on one single machine. This might be the typical day–to–day operation of this installation infrastructure when it is also used to install Linux desktop machines where the load is not very high for each server/service.

Depending on the individual performance needs, each one of these services can move to its own machine.

3.2. Scaling Up

With the way the control scripts operate, it is even possible to use more than one machine for each service.

If the clients are connected to more than one switch, use one instance of the MAC address detection script for each switch. Each instance can run on a different machine and can put detected MAC addresses in several configuration files for different DHCP servers. Each host entry in these files can have a different TFTP server entry. Finally, each pxelinux configuration file can contain a different NFS server name. The simplest way to achieve an even distribution of clients to servers is to assign the clients in a round–robin fashion to the servers.

4. Performance

For our benchmarking tests we had two servers available for serving the Linux packages to the clients, one with 100Mb/s and one with 1Gb/s network connectivity. We had 256 client machines to install.

It is pretty clear that the server with 100Mb/s connection has no chance to serve 256 clients. We therefore tried to install only 128 clients. Figure 2 shows the total network output coming from this server seen on the switch it is connected to. The link is completely saturated with about 93Mb/s throughput. The deep dip in the graph marks the end of the Linux installation and the start of our local post–installation script which uses the same server as the installation itself.

Figure 3 shows a histogram of how many clients needed what amount of time to finish their OS installation. The times are discretized in 5 second intervals. It took 45 – 47 minutes for the machines to get installed.

Note, that the graphs contain data of only 105 clients although we started the installation with 128 machines. It turns out that there are always machines

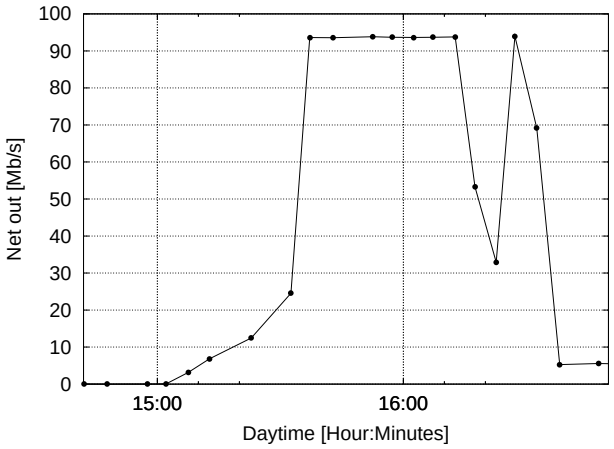

Figure 2: Network output of 100Mb/s connected NFS server during Linux installation of 105 machines.

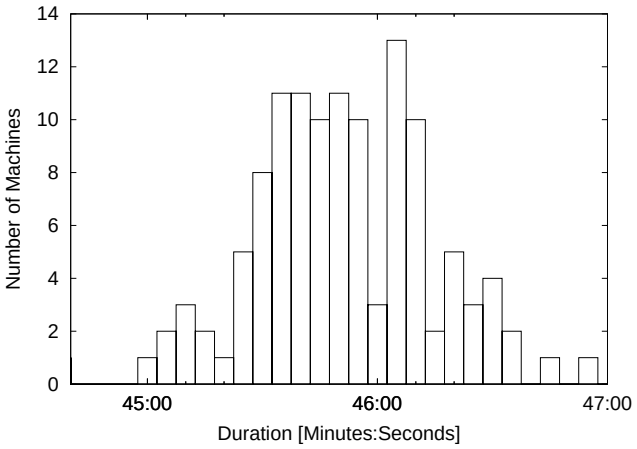

Figure 3: Distribution: Duration of Linux OS installation with 105 machine and 100Mb/s connected NFS server.

which have problems (like hardware problems) which prevent them from finishing their installation.

4.1. Estimates

Before actually doing an installation with the faster–connected server, let's first do some calculations of what we can expect by this change.

All 256 clients are connected with fast Ethernet. 256 x 100Mb/s = 25.6Gb/s. Thus, the 1Gb/s link will be 25 times oversubscribed.

Some installations by hand suggest that kickstart is not network but CPU or disk I/O bound. Using the same client machine on a 100Mb/s and a 10Mb/s Ethernet connection makes virtually no difference in installation time. Either installation takes about 10 minutes. Given the amount of 830MB data which gets installed on a machine, the following throughput calculation can be done:

$$\frac{830MB}{10min} \approx 11Mb/s$$

Since this data rate occurs on each of the 256 machines, the 1Gb/s uplink will be oversubscribed only $\frac{11Mb/s*256}{1Gb/s} = 1.4$ times. Therefore, the estimated installation time for 256 machines purely based on nominal network throughput is $1.4 \times 10min = 14min$.

It was unclear how the NFS server performs with so many clients connected to it and reading only small to medium sized files.

4.2. Hardware Configuration

The described installation infrastructure was put together to install a new farm of 256 1RU dual PIII VA1220 [11] machines. Each node is connected via a 100Mb/s fast Ethernet link to a switch. Right now all nodes are connected to the same switch.

The switch is a Cisco Catalyst 6509 with a 1Gb/s uplink connecting the farm to a multi–Gigabit Ethernet backbone.

The DHCP server and TFTP server are connected via fast Ethernet.

The NFS server used for serving the Linux packages to the clients is a Network Appliance NetApp F760 filer with a 1Gb/s Ethernet connection, 1GB of main memory, and a total of 1TB of disk space.

This set of machines is "borrowed" and not exclusively used for this Linux installations. Hence, we see some small usages for other purposes which (after the fact) turned out to be small enough to not affect the installation performance in any way.

4.3. Measurements

All 256 client machines were remotely powered on via VACM [10] within about 1 minute. They then performed a DHCP broadcast followed by three TFTP downloads and then a kickstart installation of Red Hat Linux via NFS.

Again, our clients turned out to be not very reliable with only 206 machines which actually could finish their installation.

Figure 4 shows a histogram of how many machines needed what amount of time from the last TFTP download until they finished the kickstart phase. This is the time for installing the Linux operating system on the machines. The times are discretized in 5 second intervals. The fastest machine was ready after 29:35 minutes, the slowest machine took 31:00 minutes. The average time is 30:23 minutes.

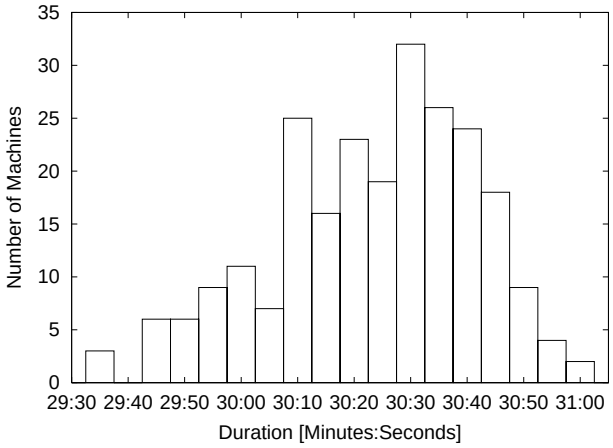

Figure 4: Distribution: Duration of Linux OS installation.

This is twice the time suggested by the network throughput calculation.

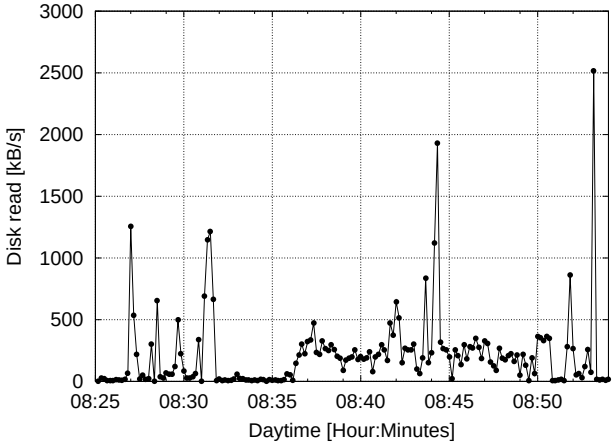

Figure 5: Disk reads on the NFS file server during Linux installation.

The disk reads on the NFS filer during the installation are very moderate as can be seen in Figure 5 (all measurements on the Network Appliance NFS server are done with the `sysstat` command with an output every 10 seconds). The entire Linux distribution of about 835MB is held in its cache (the filer has 1GB memory) so that no multiple disk accesses are necessary to serve multiple clients.

As seen in Figure 6, the CPU utilization of the NFS file server is not the reason for the slower installation time, either. The utilization is about 83% with some dips and peaks. In our all day usage of this filer, we see up to 95% - 100% CPU utilization in peak times.

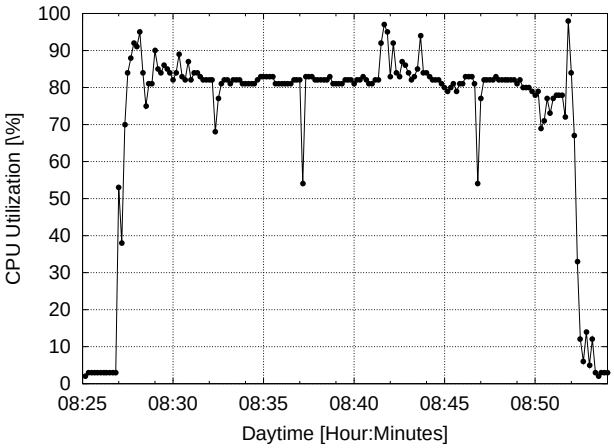

Figure 6: CPU Utilization of NFS server during Linux installation.

Figure 7 shows the actual network output seen on the NFS file server during the Linux installation. The filer is able to deliver about 370Mb/s sustained network output over its 1Gb/s link. For our usage case of 256 clients reading only small to medium sized files, this seems reasonable.

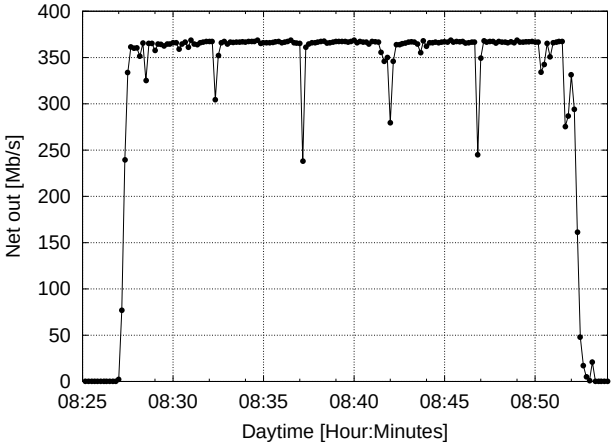

Figure 7: Network output of NFS server during Linux installation of 206 machines.

If we compare these performance numbers with the ones obtained during another test installation with only 110 clients, we see that with about half the clients connected, the filer already delivers about 370Mb/s network output (cf. Figure 8). It is not able to scale much above this rate with more clients connected. Therefore, the one NFS fileserver serving the Linux distribution during kickstart is the bottleneck in our installation! Note that this is not a limitation of the proposed infrastructure in general but of our implementation on our existing and available hardware.

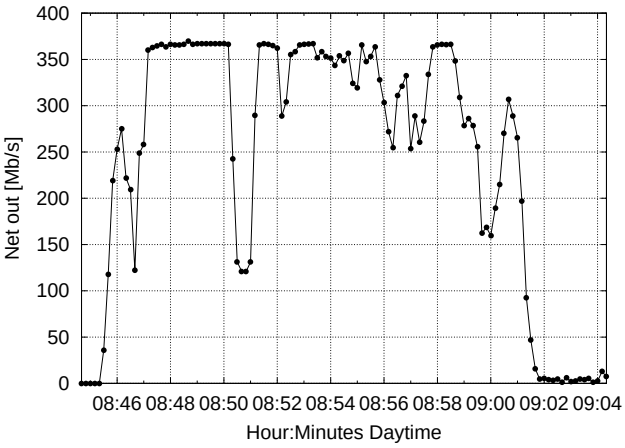

Figure 8: Network output of NFS server during Linux installation of 110 machines.

To overcome this bottleneck, you can use more than one filer and distribute the client load evenly over them. This can be done by having a different kickstart configuration file for each NFS server which is assigned (e.g. via round–robin) to the Linux machines by the PXE configuration file.

5. Further Applications

We used this infrastructure to freshly install new Linux machines. The same method can be used to re–install machines. This is necessary for system upgrades but may also be necessary when a batch job requires an operating system on a cluster of machines which is currently not stored on their hard disks. Given that a batch scheduler schedules all jobs which require this new operating system consecutively and given more fast file servers this might even be feasable.

With all the servers in place, it is now also straight forward to implement an add–on to this infrastructure for automatic and unattended desktop machine installation. One way to do this, is to design a Web page where a user supplies a MAC address and hardware configuration (disk size in particular). The backend script would add an entry to the DHCP server configuration file and would supply a suitable kickstart configuration file. The desktop machine can then be booted with a generic Red Hat Linux installation floppy.

A more general desktop installation procedure could even facilitate the full PXE functionality: a user could then choose whether s/he wants to install Linux or Windows on her/his desktop machine.

Even diskless clients (see LTSP [3]) are not difficult to support with the given installation infrastructure in place.

6. Conclusions

In this paper, we presented a general purpose high performance Linux installation infrastructure. We have implemented this infrastructure and performed the installation of a medium sized cluster of up to 256 nodes in 30 minutes.

If a shorter installation time is needed or if more clients need to be installed the presented infrastructure allows each single service (TFTP, DHCP, NFS) to be distributed over more than just one physical server. It is straight forward to modify the presented scripts to perform a round–robin load balancing over all these servers.

With these changes, the presented architecture should be capable of installing clusters of thousands of nodes as it will be necessary to do in the very near future.

Our measurements provide some data points which allow to predict roughly how to scale your installation infrastructure to your needs.

References

[1] EU DataGrid. `http://www.eu-datagrid.org`.

[2] EU DataGrid Work Package 4: Fabric Management. `http://hep-proj-grid-fabric.web.cern.ch/hep-proj-grid-fabric/`.

[3] Linux Terminal Server Project. `http://www.ltsp.org/`.

[4] H. Peter Anvin. Syslinux/pxelinux: A boot loader for linux. `http://syslinux.zytor.com/`.

[5] IBM. Linux Utility for Cluster Installation (LUI). http://oss.software.ibm.com/developerworks/projects/lui.

[6] Intel Corporation. Preboot Execution Environment (PXE) Specification Version 2.1. ftp://download.intel.com/ial/wfm/pxespec.pdf.

[7] Jean-Pierre and Remi Lefebvre. atftp. ftp://ftp.mamalinux.com/pub/atftp/.

[8] Philip M. Papadopoulos. VA Linux 1220. Talk at the "Large-Scale Cluster Computing Workshop (LCCWS) 2001", http://conferences.fnal.gov/lccws/.

[9] Philip M. Papadopoulos, Mason J. Katz, and Greg Bruno. NPACI Rocks: Tools and Techniques for Easily Deploying Manageable Linux Clusters. In *Submitted to: Proceedings of the Cluster 2001*, October 2001.

[10] VA Linux Systems, Inc. VA–Cluster Manager. http://sourceforge.net/projects/vacm/.

[11] VA Linux Systems, Inc. VA Linux 1220. http://www.valinux.com/systems/productinfo.html?product=1220.

[12] VA Linux Systems, Inc. VA Linux SystemImager Software. http://systemimager.org/.

[13] Alf Wachsmann. Howto Install Red Hat Linux via PXE and Kickstart. http://www.SLAC.Stanford.EDU/~alfw/PXE-Kickstart/.