

Implementation and Evaluation of Remote Booting and Installation for Diskless Linux Cluster Systems

Sangog Na, Hongsoog Kim, Dongsoo Han, and Chansu Yu

School of Engineering, Information and Communications University,
Yusong P.O. Box 77, Taejon, 305-600, Korea

Abstract

Diskless cluster systems have several advantages over disk attached cluster systems in which each node has its own local disks. Maintenance and upgrading of the diskless cluster system is much easier than the disk attached cluster systems. It also guarantees higher reliability since the disk drive is known as the most error prone device in a disk attached cluster system. However diskless cluster systems require additional work for bootstrapping and there are some doubts whether their performance is comparable to disk attached cluster system. In this paper we have implemented a Linux based diskless cluster system. Experience in remote booting and installation is described with the experimental results. We have used MPBench and NAS Parallel Benchmarks to evaluate and compare the performance between the diskless cluster system and disk attached cluster systems. According to our performance test, the diskless cluster system showed equivalent performance result to traditional disk attached cluster systems.

1 Introduction

Recently, cluster systems have been widely accepted as cost-effective alternatives for high performance computers [1][2]. The advantages of cluster systems over centralized systems can be summarized in terms of economics, availability and scalability. A cluster system is defined as a collection of independent computers or nodes that appear to the users as a single computer [1]. As cluster systems consists of many nodes, some software and hardware techniques have been developed to give

the illusion for users to view a collection of computing elements as a single computing resource. This is called single system image (SSI) and SSI is one of the most important features to the cluster users. However it could be a serious bottleneck at the same time for cluster systems to become popular since the related technologies are not fully developed yet.

Kai Hwang [3] identified several useful services and availability functions to be supported in SSI. These include single entry point, single file hierarchy, single control point, single virtual networking, single memory space, single job management system, and single user interface. Besides, in manager's instead of user's point of view, single maintenance and upgrading point is very important when he or she is confronted with the situation of maintaining tens or hundreds of independent PCs.

This paper discusses the implementation of remote booting of diskless cluster systems to provide more convenient single maintenance and upgrading point. In our experimental diskless cluster system, the component PCs are booted over the network, and a separate front-end server provides high capacity and highly reliable RAID (Redundant Array of Inexpensive Disks) storage through Fast Ethernet. The system is more reliable than disk attached cluster systems because the disk drive is known as the most error prone device in each PC box, and it generates more heat which raises the possibility of system faults. Thus diskless node generates less system faults, and it allows a system to be installed under harsh environment with noise and vibration. Moreover, the diskless cluster system facilitates the

maintenance of the system since one can easily upgrade or replace each node's software by changing only one copy of the software in the front-end server.

In this paper, several diskless designs are reviewed and we present the implementation of a diskless cluster system based on PXE (Pre-boot Execution Environment) [4], which was proposed for remote Linux installation or booting of diskless nodes. In spite of the advantages of diskless cluster system, it might bring serious performance degradation when there are requests for huge data movement that cannot be accommodated within a local memory of a node. Also there is no confidence whether the performance of diskless clusters is comparable to disk attached cluster systems yet. We use MPBench and NAS Parallel Benchmarks to evaluate and compare the performance between diskless cluster system and disk attached cluster system. According to our performance test, diskless cluster system showed equivalent performance result to traditional disk attached cluster systems in the test cases.

Organization of this paper is as follows. Section 2 discusses related work on diskless workstations. Section 3 presents our design for remote boot for Linux cluster systems. Performance evaluation of the diskless Linux cluster is shown in Section 4 with the concluding remarks in Section 5.

2 Related Work

Several kinds of remote booting methods are developed with different purposes. For example, X terminal was developed in order to provide educational systems at low cost while Etherboot is used to load DOS through Ethernet. The features of each method are briefly introduced in this section. operatingsystems like Linux, FreeBSD, or DOS through Ethernet. The features of the methods are briefly introduced in this section.

2.1 X Terminal

X [5] was jointly developed by MIT's

Project Athena [6] and Digital Equipment Corporation. Project Athena is a trial to integrate the new computational technology into the MIT undergraduate educational experience in 1983. X terminals are screens controlled by X server running from ROM. Font files are downloaded from X font server to RAM of the X terminal with TFTP (Trivial File Transfer Protocol) [7], and then use them. Figure 1 shows X terminal based on X window system. When X server program is changed, BOOTP (Bootstrap Protocol) [8] and TFTP can be used for downloading. The current version of X window system [9] is X11R6.4, and the current version of XFree86 [10] X window system used in Linux, is 3.3.5 based on X11R6.3pl2.

2.2 Sun Diskless Workstation

Low cost workstation solution offered by Sun Microsystems is diskless workstation where only a server has a disk and other systems use the server's disk through the network. Based on NFS (Network File System) technology, expensive hard disks can be removed or reduced in size. More specially, 'diskless workstation' means a workstation which has no disk, and 'dataless workstation' has only operating system in its local disk. In the server's disk, some amount of disk space is allocated to each client and managed by the server. Table 1 [11] compares various kinds of workstations: Stand-alone, Diskless workstation, Dataless workstation, AutoClient workstation, X terminal, and Network computer.

AutoClient systems use local disks for swap and a cache of recently used files from the / and /usr file systems. The combination of local swapping and local disk caching of frequently used files leads to a dramatic reduction in network traffic when compared to a diskless configuration. Once client is up and running, any changes made on the client are also written back to the server. If the desktop system provokes a component failure, a similarly configured system can be put in its place and brought back up in a minute.

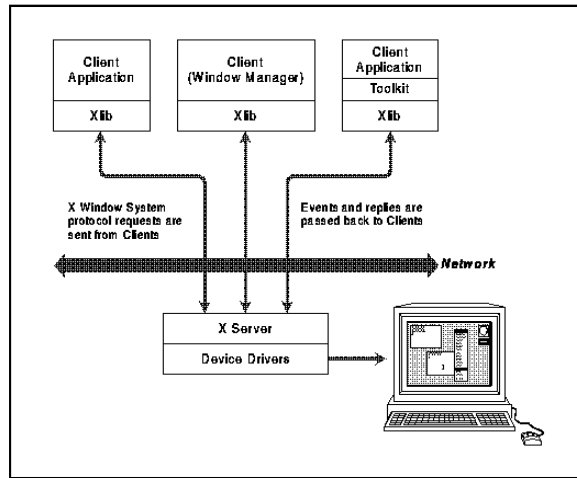


Fig. 1. X terminal

| | Configuration And Installation | Administration (Management and Software Upgrade) | Performance | Disk Usage | FRU? |
|------------------|--------------------------------|--|-------------|-------------|-------|
| Stand-alone | JumpStart ¹ | Worst | Best | Worst | Maybe |
| Diskless | Best | Best | Worst | Server-only | Yes |
| Dataless | JumpStart | Poor | Poor | Good | Maybe |
| AutoClient | Best | Best | Good/Best | Good | Yes |
| X terminal | Best | Best | Worst | N.A. | Yes |
| Network Computer | Best | Best | Unknown | N.A. | Yes |

Table 1. Comparison of workstations

| Component | Purpose | Comment |
|---|--|----------|
| Bootstrap loader | Usually in an EPROM on a NIC card, but could be put anywhere in the address space the BIOS probes in. For testing this could be put on a floppy diskette. Some booting setups may even be always run from a floppy diskette (e.g. temporary setups for testing or pedagogic purposes). | |
| BOOTP or DHCP server | For handing out IP addresses and other information when sent a MAC (Ethernet card) address. | |
| TFTP server | For sending the kernel images and other files required in the boot process. | |
| NFS server | For providing the disk partitions that will be mounted if Linux or FreeBSD is being booted. | |
| Linux or FreeBSD kernel | Has been configured to mount the root partition via NFS. | |
| Compiled-in support for mounting a swap partition via NFS | This feature is not in the stock 2.0.x or 2.2.x Linux kernels; you have to apply the contributed patches for NFS swap [13]. | Optional |

Table 2. Required components to implement Etherboot

¹ Refer to Section 2.7.

However in the case where the local disk is very slow and the network service is very fast, the performance is mainly dependent on the local disk.

2.3 Etherboot

Etherboot [12] is a free software package for booting x86 PCs over a network. In principle it works with any TCP/IP network which supports broadcasting since a diskless PC should be able to broadcast the GETIPAddress message. However, actually its usage is restricted to where the broadcasted message can be replied within a certain amount of time. Etherboot is useful for booting diskless PCs and is desirable in various situations, for example, (i) a cluster of workstations which are equally configured and maintained by a central server, (ii) a low-cost X terminal, (iii) a low cost user platform where remote disk partitions are mounted using NFS, (iv) various kinds of remote servers, e.g. a tape drive server that can be accessed with the RMT (Remote Magnetic Tape) protocol, (v) routers, and (vi) machines working in unfriendly environment to disks.

Normally Etherboot is used to load operating systems like Linux, FreeBSD or DOS. However, as the underlying protocols and boot file formats follow the agreeable industry standards, it could be used to load arbitrary images to a x86 PC. Table 2 shows the required components to implement Etherboot.

2.4 Beoboot (Rembo Technology SaRL)

Beoboot [14] is a commercial product which targets x86 PCs running Linux. Previous version developed by the same developers is Bpbatch [8] which was freely available. Beoboot works with ISC (Internet Software Consortium)² DHCPD (Dynamic Host Configuration Protocol

² **ISC is a nonprofit corporation dedicated to developing and maintaining open source reference implementations of core Internet protocols with production quality.**

Daemon) which supports DHCP extensions for PXE, which will be explained in Section 3.3. Beoboot is a bootROM-based software which is used to automatically load Linux on an arbitrary large number of Intel-based computers. A bootROM-enabled PC differs from a stand-alone PC in that when it is turned on, it asks the server for a boot program instead of looking for it on the boot sector of the hard disk. Beoboot uses this mechanism for booting Linux directly from the network server. Any preparative work is necessary for each individual client computer. This is a terrific gain when dozens or hundreds of clients have to be installed. Beoboot can be regarded as a network-based Linux loader and is particularly well suited for managing cluster of PCs working together as a supercomputer (often referred to as Beowulf systems [15]).

Beoboot software is simply installed as a daemon on a Linux network server. It will respond to the requests of client computers (according to Intel PXE standard) and send them boot files. Beoboot server is an off-the-shelf package and requires almost no configuration effort.

Although Beoboot architecture is purely network-based, it is perfectly scalable due to its multicast technology. Beoboot clients are able to share bandwidth without limitation by sharing data during file transfers. Any file transmitted from the server to an arbitrarily large number of clients is typically sent only once on the network and received by all the clients at the same time.

2.5 PXE

PXE (Pre-boot Execution Environment) is defined on the industry-standard Internet protocols and services that are widely deployed in the industry, namely TCP/IP, DHCP (Dynamic Host Configuration Protocol), and TFTP (Trivial File Transfer Protocol). These standardize the way of the interactions between clients and servers. To ensure that the meaning of the client-server interaction, it is standardized as well. To support PXE,

vendor option fields in DHCP protocol are used, which are reserved by the DHCP standard. The operations of standard DHCP and/or BOOTP (Bootstrap Protocol) servers provide IP addresses and/or NBPs (Network Bootstrap Programs) and will not be disrupted by the use of the extended protocol. Clients and servers that are aware of these extensions recognize them and use this information, and those that do not recognize the extensions ignore them.

The PXE protocol operates in the following way. The client, equipped with the PXE bootROM, initiates the protocol by broadcasting a DHCPDISCOVER message. The message contains an extension that notifies the request is coming from a client that implements the PXE protocol. If a DHCP server or a Proxy DHCP³ server implementing this extended protocol is available, the server sends the client a list of appropriate Boot Servers after several intermediate steps. The Boot Server provides the booting client with appropriate boot images for a particular boot environment. The client then discovers a Boot Server of the selected type and receives the name of an executable file on the chosen Boot Server from the DHCP server. The client uses TFTP to download the executable file from the Boot Server. Finally, the client triggers the execution of the downloaded image. At this point, the state of client must satisfy certain requirements that provide a predictable execution environment for the image. The requirements of the environment include the availability of certain areas of the client's main memory, and the availability of basic network I/O services.

³ **As the name implies, Proxy DHCP works in parallel with DHCP and provides the booting client with remote boot configuration options. Proxy DHCP provides the PXE client(s) with the following information: remote boot prompt with optional timeout, remote boot menu and PXE Boot Server discovery options.**

Note that the support of the PXE-based remote boot is required by PC98 and PC99 for all Office PCs [16] and by the Wired for Management Initiative sponsored by Intel Corp. [17]. The greatest distinguished feature of Office PC system is that it supports requirements intended to reduce TCO (Total Cost of Ownership) in the corporate environment, including the support of upgrading BIOS and remote boot capabilities. Office PC system is designed to run productivity applications, particularly in a network environment, and it is equipped with a network adapter.

PC99 [18] is the collection of the additional requirements and recommendations that make up the 1999-2000 requirements for PC system design. The Wired for Management (WfM) [19] Initiative is Intel's broad-based initiative to reduce the TCO of business computing while maintaining the power and flexibility of high-performance computing. As a result, most PC OEMs offer PXE support for LAN on motherboard platforms and for business platforms containing a PXE enabled NIC (Network Interface Card).

In addition, to upgrade existing PC platforms, PXE compliant NICs are offered by Intel [20] and 3Com [21], and many other NIC vendors. Finally, many NICs with boot ROM sockets or flash chips can be upgraded to PXE compliance. PXE compliant boot ROMs are available from Bootix Inc. [22], 3Com/Lanworks [23], and Elisa Research [24]. The PXE 2.1 specification can be found at [4].

2.6 Wake-on LAN

AMD's Magic Packet Technology [25] defines a special kind of packet which turns on a system when a LAN card of the system recognizes the broadcast packet with its MAC (Media Access Control) address. For that purpose, the power must be always supplied to the LAN card. It is called Wake-on LAN (WOL) and the corresponding source file is available in [26].

2.7 KickStart (RedHat Linux)

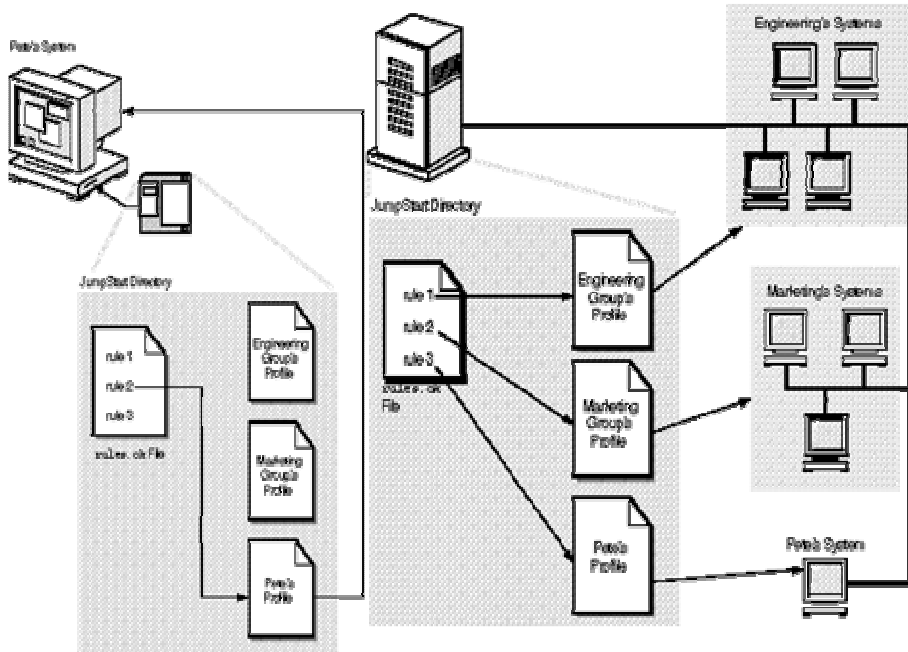


Fig. 2. JumpStart

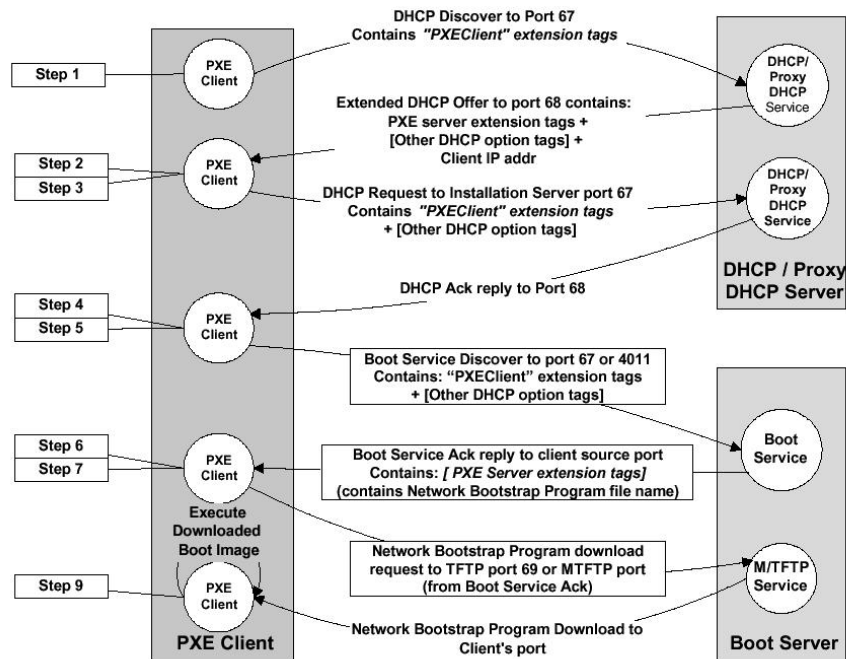


Fig. 3. PXE protocol

To install Solaris, system and network configuration information should be provided through standard input. However, JumpStart [27] can be used to avoid interactive installation. A configuration file can be placed in floppy disk or on the network as depicted in Figure 2. Similar to Sun's JumpStart installation of Solaris, it is possible to install Linux to several equally configured PCs with KickStart. It also provides a script of commands which must be executed after installation, so that it can reduce the number of rebooting operations. The installation OS image can be obtained from a CD-ROM or through the network (NFS, FTP and HTTP).

3 Remote Boot for Linux Cluster

In this section, we review two methods for remote installation and remote booting based on PXE (Pre-boot Execution Environment) technology. One is RedHat PXE daemon and the other is Beoboot of Rembo Technology SaRL. RedHat PXE is an open source based software, but Beoboot is a commercial program. Both are based on PXE protocol, and after the loading of Linux kernel, all set-up routines of the two methods are equivalent.

| Major Step | Detail Job |
|---|--|
| Change directory to kernel source directory | cd /usr/src/linux |
| Change configuration | make menuconfig Set 6 items to built-in Networking option > IP : kernel autoconfiguration and two sub-items Network device support > EtherExpressPro/100 support File systems > NFS file system support and a sub-item |
| Build kernel image | make dep; make bzImage |

Table 3. Procedure for making kernel image

3.1 Remote Boot Procedure

For remote booting, each node's bootROM BIOS must support network boot and a network card must have a ROM supporting PXE. A node is turned on either by Wake-on LAN or power switch. A node tries to boot itself through the conventional ways with floppy or hard disk. If it fails, the node with PXE capability tries to boot itself using remote files. Before downloading files from a server using TFTP (Trivial File Transfer Protocol), a node should acquire an IP address. To obtain an IP address, a node can use RARP (Reverse Address Resolution Protocol), BOOTP (Bootstrap Protocol), or DHCP (Dynamic Host Configuration Protocol). PXE uses DHCP.

After that, it tries to get NBP (Network Bootstrap Program) files using TFTP or MTFTP (Multicast Trivial File Transfer Protocol). The node executes NBP and then tries to get Linux kernel image file and initial ramdisk image file using TFTP or MTFTP. The latter is the image of ext2 file system and it is composed of several files used for booting. The node executes the downloaded Linux kernel and makes the kernel mount NFS root directory. The procedure of the PXE protocol is summarized in Figure 3.

Two image files are required for remote booting as depicted in the Figure. One is the kernel image and the other is the ramdisk image. Table 3 shows the execution procedure for making the kernel image. Initial Ramdisk, or called initrd, is the image of ext2 file system but it is compressed using gzip. Usually, it contains loadable modules and these modules are used when network device drivers are loaded dynamically.

3.2 PXE/Beoboot Server

3.2.1 RedHat PXE Daemon

The set-up procedure for PXE server whose IP address is 10.10.10.1 is described in Table 4.

3.2.2 Beoboot

| Major Step | Sub Step | Command | |
|--|--|---|--|
| Configure to start pxe at boot time using ntsysv | | | |
| Create directory /tftpboot/X86PC/UNDI/linux-boot | | | |
| Create bstrap.0 and linux.0 | Download from ftp://ftp.redhat.com/rawhide/SRPM S/SRPMS/pxe-0.1.14.src.rpm | | |
| | Install rpm | rpm -ihv pxe-0.1.14.src.rpm | |
| | Change directory to directory installed | cd /usr/src/redhat/SOURCES | |
| | Extract pxe source files | tar xvfz pxe-linux.tar.gz | |
| | Patch the source of PXE | patch -p0 <pxe-linux-config.patch | |
| | | patch -p0 <pxe-1.0-cmdlinearg.patch | |
| | Update bstrap.c and linux.c for | cd pxe-linux/nbp/linux | |
| | | Comment out a block containing type=do_menu() and insert type=14; in bstrap.c(for skipping menu select) | |
| | | In linux.c, change "ks console=ttyS0,115200" into "rw root=/dev/nfs console=ttyS0,19200" (for nfs root) | |
| | Build NBPs | make | |
| Copy files to TFTP directory. | cp bstrap.0 /tftpboot/X86PC/UNDI/Bstrap | | |
| | cp linux.0 /tftpboot/X86PC/UNDI/linux-boot | | |
| Copy kernel image file to TFTP directory | cp arch/i386/boot/bzImage /tftpboot/X86PC/UNDI/linux-boot/linux.1 | | |
| Modify PXE configuration | network interface : eth0 -> eth1 | | |
| | Remove comment lines related to linux-boot | | |

Table 4. Set-up procedure for PXE server

| Major Step | Detail Job |
|---|--|
| Install Beoboot package using tar | |
| Copy kernel image file to files/ directory | |
| Copy initial ramdisk image file to files/ directory | |
| Update Beoboot configuration file | Change LinuxArgs to "rw root=/dev/nfs console=ttyS0,19200" |
| | Change BaseDir to the installed directory |
| Execute beoboot | |

Table 5. Set-up procedure for Beoboot server

| Major Step | Target objects or detail procedure |
|---|---|
| Copy files from another machine using tar | etc, dev, bin, home, lib, sbin, tmp, var |
| Make directories | boot, proc, root, usr |
| Make `fastboot' file and set to non-removable | chattr +i fastboot |
| Generate symbolic link at /tftpboot | <pre>ln -s /abc/export/node* /tftpboot cd /tftpboot csh @ i=0 while (\$i<24) if (\$i<10) then set id=0\$i else set id=\$i endif ln -s node\$id 10.10.10.1\$id @ i++ end</pre> |
| Export /usr to node* with read-only | |
| Export /user to node* with read-write | |
| Update files | etc/HOSTNAME etc/sysconfig/network etc/sysconfig/network- scripts/ifcfg-eth0 etc/inittab etc/fstab |
| Delete files from etc/rc.d/rc6.d and etc/rc.d/rc0.d | K50inet, K75netfs, K90network, S00killall |

Table 6. Set-up procedure for NFS server

Table 5 shows the set-up procedure for Beoboot server.

3.3 NFS, DHCP Server

Table 6 shows the set-up procedure of NFS server [28] whose IP address is 10.10.10.1. According to the PXE's boot sequence, The IP address of a node can be obtained using DHCP. Therefore, DHCP server must be prepared beforehand. ISC DHCP supporting PXE extension can be downloaded from [29],

after installing DHCP server, the following line has to be appended to the configuration file:

```
option vendor-class-identifier
"PXEClient";
```

The line gives information to the PXE client on how to interrogate the Proxy DHCP service because the DHCPPOFFER from the DHCP service contains an Option #60 "PXEClient" tag without corresponding Option #43 tags or a boot file name. Table 7 shows the set-up procedure for DHCP server.

| Major Step | Detail procedure |
|--|--|
| Create /etc/dhcpd.conf(refer to Appendix A,2) | |
| Using ntsysv, set to run dhcpd at boot time | |
| For the first time to run dhcpd, file /var/state/dhcp/dhcpd.leases does not exist, and then you get a error message. | touch /var/state/dhcp/dhcpd.leases /etc/rc.d/init.d/dhcpd start |

Table 7. Set-up procedure for DHCP server

3.4 Remote Installation

After the initial remote booting, the whole operating system has to be installed, which usually requires interactive operations. Many nodes have to be monitored and interactive key stroke is required for each node. In Sun's Solaris, it is possible to install nodes automatically using JumpStart, and RedHat Linux uses KickStart [30] for automatic installation as explained in Section 2.7. In RedHat Linux 6.1, PXE and KickStart are provided to support remote installation. Table 8 shows components for remote installation.

4 Performance Evaluation

Real implementation of a diskless cluster system and the corresponding performance study is presented in this section. Section 4.1 describes the system configuration and Section 4.2 is devoted to performance evaluation of the diskless cluster system. Remote boot time is one of the most related measures in diskless

cluster system. However, the general computing performance of the diskless cluster system is also tested to ensure that the overall performance is not degraded due to the absence of local disks.

4.1 ABC (A Board Cluster) Configuration

ABC or A Board Cluster is an experimental diskless cluster system being built at ICU supported by Korea Institute of Space Telecommunication. The main purpose of this cluster system is to run embarrassingly parallel applications [31] with no high volume data exchange. Configuration of ABC is presented in Figure 5 and Table 9 shows the specification of the system.

4.1.1 Network Configuration

IP addresses 10.10.10.x are reserved for private network addresses as mentioned in [15] and [32]. Among them, IP addresses from 10.10.10.100 to 10.10.10.123 are assigned to Client nodes and IP addresses 10.10.10.50~52 are used for three power control units, by which we can remotely control, i.e. turn on, off, or reset the client nodes. Each of the two Fast Ethernet switch hubs is connected with twelve client nodes. We connect the server to one switch hub and then to another switch hub⁴.

4.1.2 Software Architecture

ABC system has several design goals. High availability, high reliability, easy to use and single system image are the major goals that ABC system pursues. Separate software modules are developed or installed from the packages available to achieve these goals. Especially for the

⁴ **The connecting order can be a switch hub, server, and another switch hub. We haven't compared the performance of the two cases but we chose the previous connecting order because it is easier to assemble.**

support of single system image, we have developed Cluster Management System on top of the software modules. Monitoring and administration operation for ABC system can be performed on the Cluster Management System. CMS user interface is developed on the Web browser. Thus remote access to the CMS through the Internet is also possible. Figure 6 shows the software architecture of ABC system. PBS (Portable Batch System) [33] is installed for job management and MPICH/LAMMPI is installed for MPI (Message Passing Interface) library.

4.1.3 Serial Console

Since each client node does not have its own display and keyboard, two serial hubs are used to redirect consoles. Serial port of each client node is connected to a Rocketport serial Hub, and its speed is set with the baud rate of 19200, which must be set in ROM BIOS. In order to make the kernel's default baud rate to 19200, we have implemented it in a file `drivers/char/serial.c`.

4.1.4 Miscellaneous

To support software power control in off line state of diskless nodes, 3 master switches of APC Inc. are installed where each master switch covers 8 nodes.

4.2 Experiment Setup & Results

In this section, we present experiment results with the ABC diskless cluster system. Firstly, remote boot times in various circumstances are presented.

A front-end server with RAID provides all the files required to boot all the client nodes, and thus it could be a communication bottleneck. Note that in disk attached cluster systems, the booting time is almost equivalent to that of a node. Secondly, general computing performance is measured using MPBench [34] and NAS Parallel Benchmarks [35].

4.2.1 Remote Boot Time

For the measurement of remote boot time, we measured the time from turning on the system to the completion of the execution of all the daemons of nodes. Server writes calendar time on a file, makes one node be turned on using Wake-on LAN, and in the last boot step, /etc/rc.d/rc.local.mine of the booted node writes the server's calendar time using rsh (remote shell). It takes about 90 seconds for one node.

When all 24 client systems are turned on simultaneously, some nodes are not booted because of network packet collision. When client systems are turned on in sequence every 20 seconds, it takes about 9 minutes (90 seconds + 23 * 20 seconds = 550 seconds) to complete the remote booting. Remote boot time can be reduced by booting several nodes at the same time. The maximum number of nodes to boot simultaneously is checked, and the stable value is around 4.

Four systems are turned on at the same time, and after 40 seconds, other 4 systems are turned on in order. The total boot time has been taken about 5 minutes (90 seconds + 5 * 40 seconds = 290 seconds) in that condition. Table 10 summarizes experimental results of remote boot of ABC system in various circumstances.

In using Wake-on LAN to wake up client nodes with Linux SMP, there is a major drawback in turning off the systems. Every system's switch should be manually turned off for Wake-on LAN. As EMP (Emergency Management Port) port can be used for power control, it is possible to control power such as reset or turning off with Intel serverboard's BMC (Baseboard Management Controller) [36]. doghalt.c in Appendix A.3 uses BMC, and turns off the system after 30 seconds.

4.2.2 MPBench

MPBench is a benchmark program to evaluate the MPI and PVM performance

on clusters. Table 11 shows the list of what can be measured using MPBench.

Figure 8 through Figure 11 compare the MPBench results of LLCBench [34] running on ABC cluster system versus those on another disk attached cluster system whose configuration is shown in Table 12. MPICH version 1.2.0 [37], one of MPI (Message Passing Interface)

| Group size (Number of nodes to start booting at the same time) | Delay (Delay time between successive bootings of groups) | Result (Time to boot all client nodes) |
|---|---|---|
| 1 | 20 seconds | 551 seconds |
| | 15 seconds | 434 seconds |
| | 10 seconds | 386 seconds |
| | 8 seconds | 338 seconds |
| | 0 second ⁵ | Fail (10 nodes fail) |
| 2 | 40 seconds | 532 seconds |
| | 30 seconds | 421 seconds |
| | 25 seconds | 419 seconds |
| 3 | 40 seconds | 434 seconds |
| | 30 seconds | 350 seconds |
| | 25 seconds | Fail (21 nodes fail) |
| 4 | 50 seconds | 341 seconds |
| | 45 seconds | 315 seconds |
| | 40 seconds | 291 seconds |
| 5 | 45 seconds | 335 seconds |
| | 40 seconds | 316 seconds |
| 6 | 70 seconds | 366 seconds |
| | 60 seconds | Fail (6 nodes fail) |
| | 50 seconds | Fail (13 nodes fail) |
| 7 | 70 seconds | 300 seconds |
| | 60 seconds | Fail (7 nodes fail) |

Table 10. Remote boot time

⁵ Delay of zero second means all 24 client nodes are tried to boot altogether.

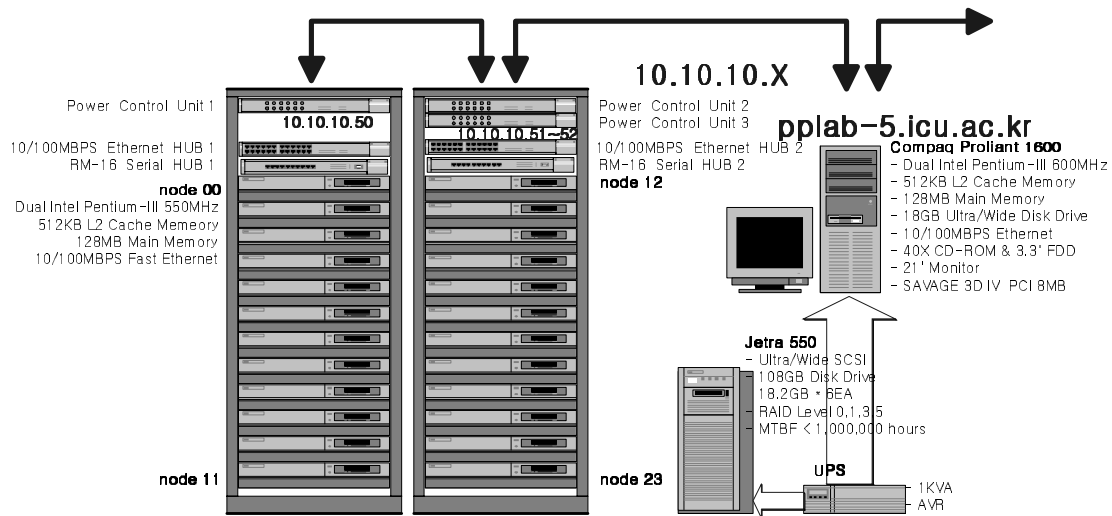


Fig. 5. ABC Configuration

| | | |
|-------------------|--------------------|--|
| Server Node | kinds of machine | Proliant 1600 by Compaq |
| | CPU | Dual Intel Pentium-III 600MHz (512KB L2 Cache) |
| | Memory | 128 MB |
| | Hard disk | 18 GB |
| | RAID | 108 GB (RAID-5) |
| | OS | Alzza Linux 6.1 by LinuxOne |
| Client Nodes (24) | kinds of machine | Intel L440GX+ mainboard |
| | CPU | Dual Intel Pentium-III 550MHz (512KB L2 Cache) |
| | Memory | 128 MB |
| | Hard disk | no floppy, no hard disk, no CD-ROM |
| | OS | Alzza Linux 6.1 by LinuxOne |
| Misc. | FastEthernet HUB | Superstack II 10/100Mbps 24-port 2EA by 3Com |
| | Serial HUB | RocketPort (16-port) 2EA by Control |
| | Power control unit | Master switch AP9212 8-port 3EA by APC |

Table 9. ABC specification

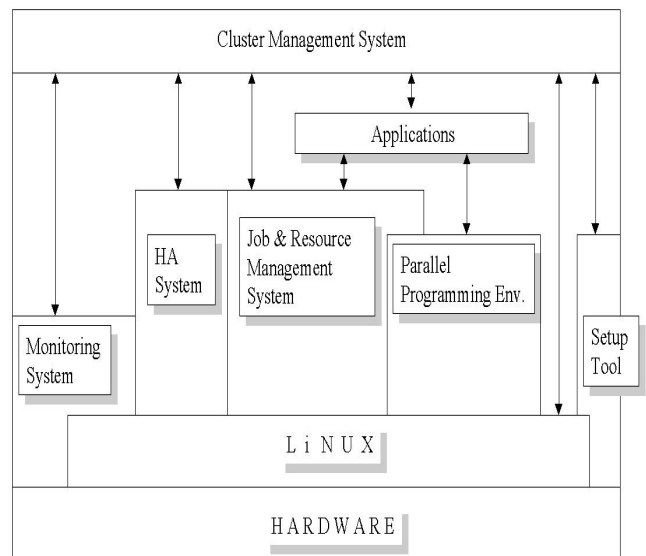


Fig. 6. Structure of Cluster Management System

| | |
|--|------------------------|
| MPI_Send()/MPI_Recv() Bandwidth | Kb/second vs. bytes |
| MPI_Send()/MPI_Recv() Application latency or Gap time | μ s vs. bytes |
| MPI_Send()/MPI_Recv() Roundtrip or 2 * Latency | turns/second vs. bytes |
| MPI_Send()/MPI_Recv() Bidirectional Bandwidth | Kb/second vs. bytes |
| MPI_Bcast() broadcast | Kb/second vs. bytes |
| MPI_Reduce() reduction (sum) | Kb/second vs. bytes |
| MPI_AllReduce() reduction (sum) | Kb/second vs. bytes |
| MPI_Alltoall() Each process sends to every other process | Kb/sec vs. bytes |

Table 11. MPBench measurement list

| | |
|------------|--|
| CPU | Pentium III 500MHz (L2 cache 512K) * 2 |
| Memory | 256M Byte |
| Hard disk | EIDE 9G |
| Main board | Intel L440GX+ |

Table 12. Node's configuration

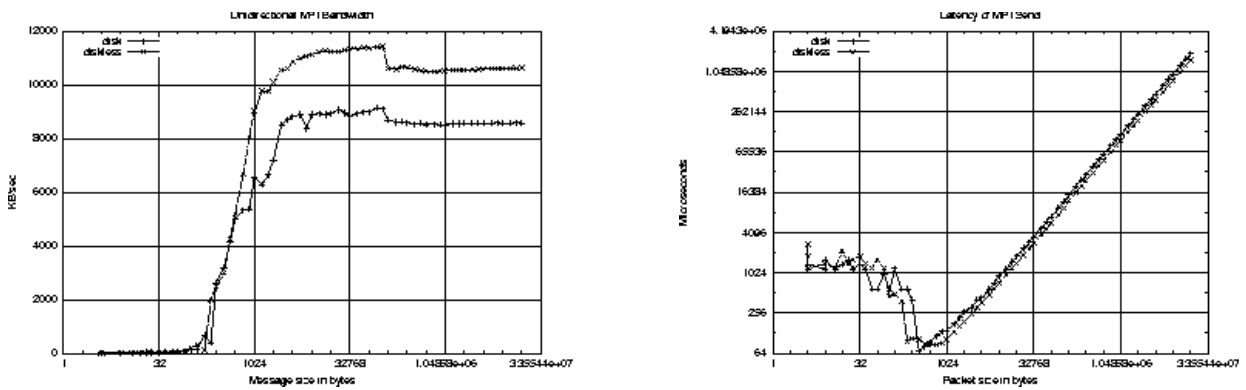


Fig. 8. Unidirectional MPI Bandwidth(left) and Latency of MPI Send(right)

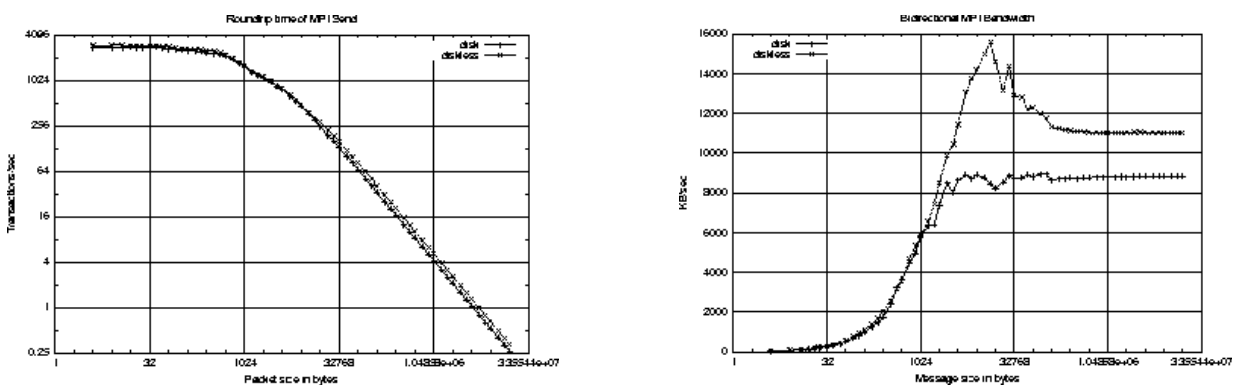


Fig. 9. Roundtrip time of MPI Send(left) and Bidirectional MPI Bandwidth(right)

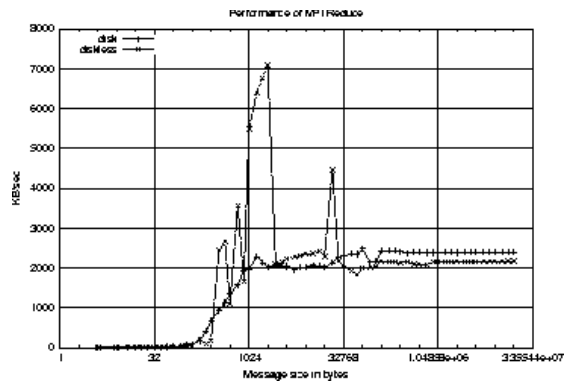
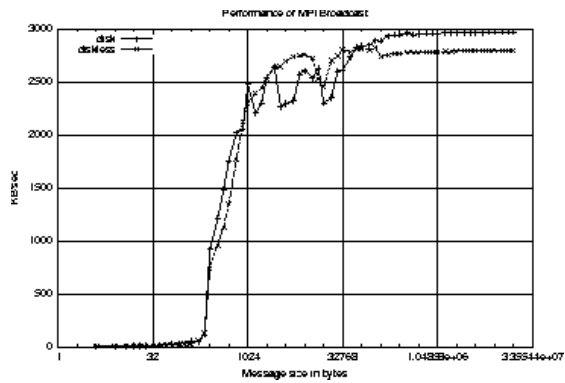


Fig. 10. MPI Broadcast(left) and MPI Reduce(right)

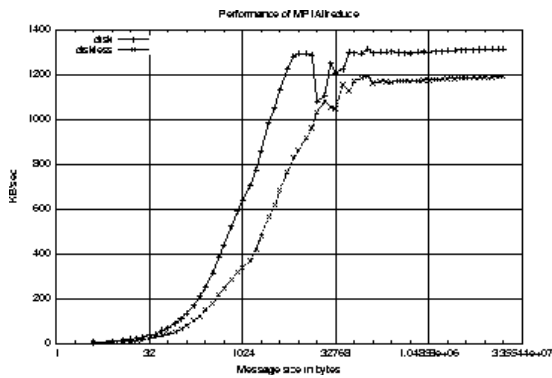


Fig. 11. MPI Allreduce

| Benchmark Code | Class A | Class B | Class C |
|-------------------------------|--------------------|--------------------|----------|
| Embarassingly parallel (EP) | 2^{28} | 2^{30} | 2^{32} |
| Block tridiagonal solver (BT) | 64^3 | 102^3 | 162^3 |
| Multigrid (MG) | 256^3 | 256^3 | 512^3 |
| Conjugate gradient (CG) | 14000 | 75000 | 150000 |
| 3-D FFT PDE (FT) | $256^2 \times 128$ | 512×256^2 | 512^3 |
| Integer sort (IS) | 2^{23} | 2^{25} | 2^{27} |
| LU solver (LU) | 64^3 | 102^3 | 162^3 |
| Pentadiagonal solver (SP) | 64^3 | 102^3 | 162^3 |

Table 13. NAS Parallel Benchmark problem size

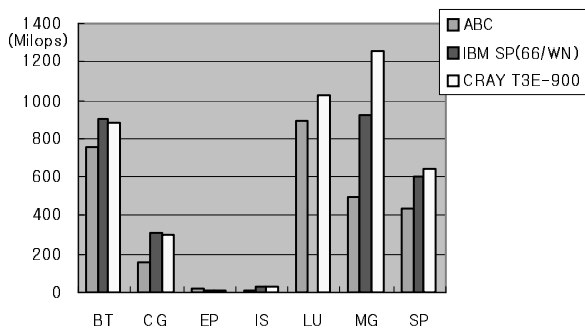


Fig. 12. Performance results on NAS Parallel benchmarks

| | ABC | IBM SP(66/WN) | CRAY T3E-900 |
|----------|--------|---------------|--------------|
| BT(A,16) | 756.02 | 899.9 | 879.2 |
| CG(A,16) | 155.23 | 314.2 | 299.3 |
| EP(A,16) | 18.87 | 10.5 | 12.9 |
| IS(A,16) | 9.4 | 29.7 | 35 |
| LU(A,16) | 895.54 | | 1022.2 |
| MG(A,16) | 495.97 | 923.4 | 1255.6 |
| SP(A,16) | 436.19 | 606.0 | 643.0 |

Table 14. Result of NPB Class A

implementations, is used for this benchmark evaluation.

Latency of MPI Send and roundtrip time of MPI Send (Figure 9), performance of MPI Broadcast and performance of MPI Reduce (Figure 10) show almost the same performance in both cases. In case of unidirectional MPI Bandwidth (Figure 8) and bidirectional MPI Bandwidth (Figure 9), ABC cluster shows slightly better performance. It is due to the difference in processor speed and extra works running in the cluster system with local disks. ABC cluster suffers slightly from the MPI Allreduce test (Figure 11)). This is because that smaller number of nodes(7 nodes instead of 16 nodes) are used in the ABC cluster.

4.2.3 NAS Parallel Benchmarks

BT, CG, EP, IS, LU, MG, and SP benchmark programs are evaluated in the system. Table 13 illustrates the description and problem size of NAS Parallel Benchmarks. Table 14 and Figure 12 shows the comparison result of the ABC performance for NAS Parallel Class B benchmark programs with that of IBM SP and CRAY T3E-900 machines with 16 processors. The number in parenthesis represents the number of processes. Although ABC shows slightly better performance in EP, in most cases the performance result of ABC is worse than that of IBM SP and CRAY T3E-900 machines. However the performance gap is not so remarkable. Thus when we take into account of the cost-effectiveness of system, we can conclude that ABC system showed promising performance results in the cases.

5 Conclusion

Implementation and experimentation of remote booting/installation on diskless Linux cluster system is described in this thesis. Our performance study using MPBench and NAS Parallel Benchmarks shows that diskless cluster system performs on par with the traditional cluster systems with local disks. Without degrading the performance, MTBF (mean

time between failure) is greatly improved because this system has removed the most error prone units in each PC box. The diskless cluster system also provides easier backup and maintenance handles because they can be operated on the server. However, in order for the diskless cluster system to be successful, the server node must be highly reliable and efficient, and each node's network driver must be stable.

References

- [1] Gregory F. Pfister. In Search of Clusters. Pentice Hall PTR, 1998.
- [2] Rajkumar Buyya. High Performance Cluster Computing: Architectures and Systems. Prentice Hall PTR, 1999.
- [3] Kai Hwang, Hai Jin, Edward Chow, Cho-Li Wang, and Zhiwei Xu. Designing SSI clusters with Hierarchical Checkpointing and Single I/O Space. IEEE Concurrency, January-March 1999.
- [4] Intel. Preboot Execution Environment(PXE) Specification, Ver. 2.1, September 1999.
- [5] Volume One: Xlib Programming Manual for Version 11 of the X Window System. O'Reilly & Associates, Inc.
- [6] G. A. Champine, D. E. Geer, W. N. Ruh. Project Athena as a distributed computer system. IEEE Computer, Volume 23, Issue 9, pp. 40-51, September 1999.
- [7] RFC 1350 : THE TFTP PROTOCOL(REVISION 2), July 1992.
- [8] RFC 951 : BOOTSTRAP PROTOCOL(BOOTP), September 1985.
- [9] Robert W. Scheifler, James Gettys, Al Mento, Donna Converse. X Window System: Core and Extension Protocols : X Version 11, Releases 6 and 6.1. Digital Press, Feb. 1997.
- [10] XFree86, <http://www.xfree86.org>.
- [11] Sun Microsystems. Solstice TM Autoclient TM v2.0.
- [12] Robert Nemkin, Al Dev, Markus Gutschke, and Ken Yap. Diskless Nodes HOW-TO document for Linux. September 1999.
- [13] Swapping via NFS for Linux. <http://www.instmath.rwth-aachen.de/heine/nfs-swap/nfs-swap.html>.
- [14] Beoboot.

<http://www.rembo.com/beoboot/>.

[15] Thomas L. Sterling, John Salmon, Donald J. Becker, and Daniel F. Savarese. How to Build a Beowulf. The MIT Press, 1999.

[16] PC99 for all Office PCs. <http://www.pcdesguide.com/pc99>.

[17] Intel. Wired for Management Baseline, Ver. 2.0 Release Dec. 18, 1998.

[18] PC DESIGN GUIDE. <http://www.pcdesguide.org/>.

[19] Wired for Management. <http://developer.intel.com/ial/WfM/>.

[20] Intel PRO/100+ Management Adapter. <http://www.intel.com/network/products/pro100mgmt.htm>.

[21] EtherLink 10/100 PCI Managed. <http://www.3com.com/products/nics/3c905c.html>.

[22] Bootix Inc. <http://www.bootix.com>.

[23] Managed PC Boot Agent. <http://www.3com.com/products/dsheets/400350.html>.

[24] Elisa Research. <http://www.elisaresearch.com/>.

[25] Magic Packet Technology White Paper, November 1995. <http://www.amd.com/products/npd/techdocs/20213.pdf>.

[26] `ether-wake.c`. <ftp://cesdis.gsfc.nasa.gov/pub/linux/misc/ether-wake.c>.

[27] Paul Anthony Kasper, and Alan McClellan. Automating Solaris Installations: A Custom JumpStart Guide. Prentice Hall PTR (ECS Professional), March 1995.

[28] Ofer Maor. NFS-Root-Client Mini-Howto, February 1999. <http://www.linuxdoc.org/HOWTO/mini/NFS-Root-Client.html>.
0Gflops for \$150k.

[29] ISC DHCP Distribution. <ftp://ftp.isc.org/isc/dhcp/dhcp-3.0b1p113.tar.gz>.

[30] Martin Hamilton. RedHat Linux KickStart HOWTO, January 1999. <http://www.linuxdoc.org/HOWTO/KickStart-HOWTO.html>.

[31] Barry Wilkinson, and Michael Allen. Parallel Programming. Prentice Hall, 1999.

[32] Bill Saphir, Patrick Bozeman, Remy Evard, and Pete Beckman. Production Linux Clusters. In SC'99 Tutorial, November 1999.

[33] R. Henderson, D. Tweten. "Portable Batch System: Requirements Specification," NAS, NASA Ames Research Center, April 1995.

[34] Philip J. Mucci. LLCbench (Low level Characterization Benchmarks). <http://icl.cs.utk.edu/projects/llcbench/>.

[35] D. Bailey, T. Harris, W. Saphir, R. van der Wijngaart, A. Woo, and M. Yarrow. "The NAS Parallel Benchmarks 2.0," Technical Report NAS-95-020, NASA Ames Research Center, December 1995.

[36] Intel. Intelligent Platform Management Interface Specification ver. 1.0, November 1999.

[37] W. Gropp, E. Lusk, N. Doss, A. Skjellum, "A high-performance, portable implementation of the MPI message passing interface standard," Parallel Computing, Vol. 22, No. 6, pp. 989-828, Sep. 1996.

[38] KAIST GALAXY Performance Benchmark. <http://nurapt.kaist.ac.kr/galaxy/benchmark.html>.

[39] Michael S. Warren, Timothy C. Germann, Peter S. Lomdahl, David M. Beazley, and John K. Salmon. Avalon: An Alpha/Linux Cluster Achieves 1